

SERIE . A
N° D'ORDRE 201
N° DE SÉRIE : 22

THESE

présentée devant

L'UNIVERSITÉ DE RENNES

U.E.R. Mathématiques et Informatique

pour obtenir

Le grade de Docteur ès-Sciences Mathématiques

par

Jacques LENFANT

Comportement des programmes dans leur espace d'adresse.
Application à la gestion des mémoires hiérarchisées.

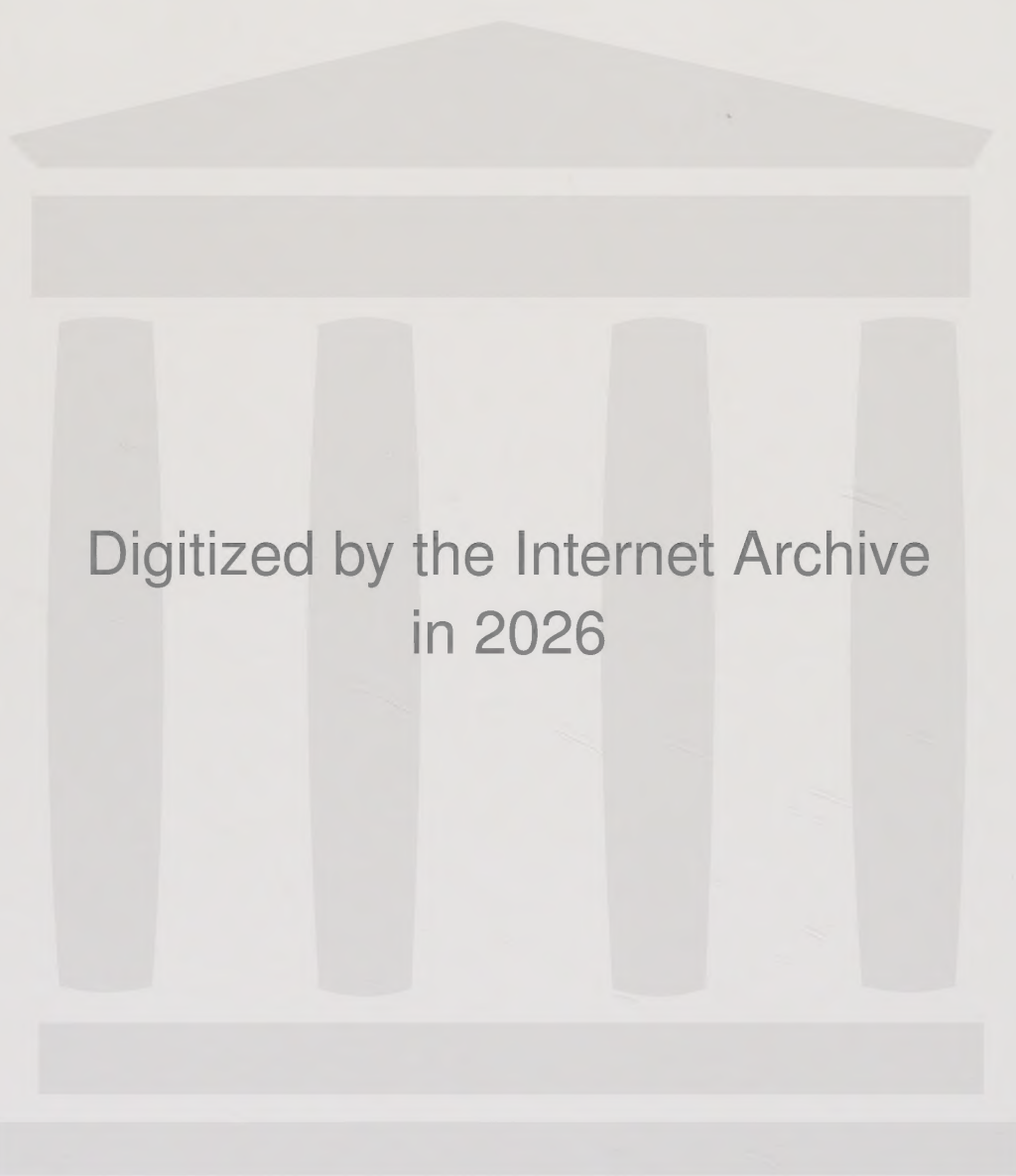
Soutenue le 14 Décembre 1974 devant la Commission d'Examen

MM. M. METIVIER

A. BENSOUSSAN
E. COFFMAN
E. GELENBE
C. PAIR
J.P. VERJUS

Président

Examineurs



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41553611_0004

THESE

présentée devant

L'UNIVERSITÉ DE RENNES

U.E.R. Mathématiques et Informatique

pour obtenir

Le grade de Docteur ès-Sciences Mathématiques

par

Jacques LENFANT

Comportement des programmes dans leur espace d'adresse.
Application à la gestion des mémoires hiérarchisées.

Soutenue le 14 Décembre 1974 devant la Commission d'Examen

MM. M. METIVIER

Président

A. BENSOUSSAN

E. COFFMAN

E. GELENBE

C. PAIR

J.P. VERJUS

Examineurs



UNIVERSITE DE RENNES

.....

U.E.R. SCIENCES ET PHILOSOPHIE

.....

Doyens Honoraires

M. MILON Y.
M. TREHIN R.
M. SCHMITT M.
M. LE MOAL H.
M. MARTIN Y.
M. BOCLE J.

Professeurs Honoraires

M. FREYMANN R.
M. HAGENE P.
M. ROHMER R.
M. TREHIN R.
M. SALMON - LEGAGNEUR F.
M. des ABBAYES H.
M. MILON Y.

Maîtres de Conférences Honoraires

M. GRILLET L.
M. LECRECO A.

Mathématiques

Professeurs

M. GUERINDON Jean
M. METIVIER Michel
M. GIORGIUTTI Italo
M. TOUGERON Jean-Claude
M. MIGNOT Alain
M. HOUDEBINE Jean
M. KEANE Michaël
M. BOULAYE Guy

Maîtres de Conférences & Chargés de Cours

M. BERTHELOT Pierre
M. BREEN Lawrence
M. CAMUS Jacques
M. CONZE Pierre
M. FERRAND Daniel
M. GUIVARC'H Yves
M. LE GOUPIL Jean
M. LERMAN Israël
M. NEDELEC Jean-Claude
M. TRILLING Laurent
M. VERJUS Jean-Pierre

Physique

M. VACHER Michel
M. VIGNERON Léopold
M. LE BOT Jean
Mme ROBIN née SALOMOND
M. ROBIN Stéphane
M. LE MONTAGNER Serge
M. MEVEL Jean-Yves
M. MEINNEL Jean
M. REGENSTREIF Edouard
M. BRUN Pierre
M. LE ROUX Emile
M. DUBOST Gérard
M. ARQUES Pierre-Yves
M. HAEUSLER Claude
M. NUSIMOVICI Michel

M. GUIDINI J.
M. DECAMPS E.A.

Professeurs

Maîtres de Conférences & Chargés de Cours

Chimie

M. VENE Jean
M. LE MOAL Henri
M. LEVAS E.
M. PELTIER Daniel
M. PRIGENT J.
M. FOUCAUD André
M. LANG Jean
M. CARRIE R.
M. GUERILLOT Claude-Roger
M. KERFANTO Michel (E.N.S.C.R.)
M. DABARD René
M. GRANDJEAN Daniel
M. LUCAS Jacques
M. MARTIN Guy

M. TALLEC André

Chimie Biologique

M. JOUAN Pierre
M. DUVAL Jacques

Géologie

M. COGNE Jean
Mlle DURAND Suzanne
M. BOILLOT Gilbert

M. HAMEURT Jean-Marie
M. LARDEUX Hubert

Zoologie

M. RICHARD Gaston
M. MAILLET Pierre
M. RAZET Pierre
M. FOLLIOU Roger
M. ALLEGRET Paul
M. MANIEY Jacques
M. JOLY Jean-Marie
M. LEFEUVRE J.C.

M. CALLEC Jean-Jacques

Botanique

M. VILLERET Serge
M. CLAUSTRES Georges
Mlle GOAS Gabrielle
Mlle GOAS Marie
Mme LEMOINE Cécile

M. TOUFFET Jean

Philosophie

M. ORTIGUES Edmond

M. JACQUES Francis
M. BEYSSADE Jean-Marie

Je tiens à exprimer ma profonde gratitude à Monsieur le Professeur GELENBE qui a orienté mes recherches et m'a constamment prodigué conseils et encouragements.

Je remercie Monsieur le Professeur METIVIER qui a apporté son soutien aux recherches en modélisation et me fait l'honneur de présider le jury de thèse.

Je remercie très particulièrement Monsieur le Professeur VERJUS qui m'a accueilli dans son laboratoire et a favorisé la naissance de ce domaine de recherche.

Je remercie Monsieur le Professeur E. COFFMAN dont les remarques ont été un encouragement très précieux.

Je suis reconnaissant à Monsieur le Professeur PAIR et Monsieur le Professeur BENSOUSSAN qui, en dépit de leurs lourdes tâches, ont bien voulu s'intéresser à cette thèse et participer au jury.

Les travaux présentés dans cette thèse doivent beaucoup à l'excellente ambiance de travail du groupe "Evaluation de Systèmes" que dirige E. GELENBE au Laboria. Que l'occasion me soit donnée de rendre un hommage particulier à D. POTIER et J. LEROUDIER.

Je remercie également P. BURGEVIN, Ingénieur chercheur à l'université de RENNES, dont le concours me fût précieux.

Enfin, je remercie Madame LEBAIL, Mesdemoiselles GLEVEO, NOGUES et ROMAN pour la réalisation matérielle de cette thèse.

Certains des travaux présentés dans cette thèse, en particulier ceux qui font l'objet du chapitre 5, ont été financés par le contrat de recherche 74.003 accordé par le Service de Synthèse et d'Orientation de la Recherche en Informatique (SESORI - IRIA).

AVANT PROPOS

Elément essentiel pour la conception des systèmes informatiques, le comportement des programmeurs reste fort mal connu. A la variété des possibilités de l'outil répond la diversité des attitudes de l'utilisateur. Définir précisément ces dernières pour adapter la machine est insuffisant car on ne peut ignorer la réaction. Comme le notait Wilkes à propos d'expérience sur le système CTSS :

.... the design of the scheduling algorithm can influence profoundly the behaviour of users and by making the system attractive for some types of work rather others, determine the use that will be made of the system.

Le temps de réflexion à la console, la fréquence d'utilisation des périphériques ou des services offerts par le moniteur ont pu être rapprochés statistiquement d'autres facteurs tels que l'expérience des programmeurs (travaux de J. Leroudier, de J.C. Le Faou). Nous nous intéressons ici à la dispersion des références dans l'espace d'adresse. La nature des algorithmes mis en oeuvre, le choix des structures de données, l'utilisation des règles de programmation plus ou moins élaborées, la structure du calculateur, la connaissance par l'utilisateur des caractéristiques du système sont autant d'éléments qui influent sur la dispersion des références. En paramétrant le comportement des usagers, les modèles de comportement de programme permettent l'adaptation à chaque catégorie d'utilisateur des résultats obtenus sur les taux de défauts de page, la taille des localités..... Cette démarche suppose que l'on puisse déterminer des valeurs typiques des paramètres des modèles à partir de programmes réels.

L'utilisation des modèles de comportement de programme est inévitable pour l'étude analytique de modèles de systèmes informatiques. Cette approche est également très intéressante pour les études par simulation. Dans ce dernier cas une alternative est possible : alimenter le modèle avec des chaînes de références relevées pendant l'exécution de programmes réels. Une question se pose alors : jusqu'à quel point

ces chaînes réelles - mais ne pouvant représenter plus de quelques secondes d'exécution - sont elles typiques ? ce problème n'est guère différent de celui de la justification des modèles de comportement. Notons un inconvénient de l'utilisation des chaînes de références "réelles" : il est difficile d'en tirer des renseignements sur l'influence de modifications du comportement des programmes. En revanche, les modèles stochastiques de programme sont l'équivalent, pour l'analyse des systèmes et leur simulation des programmes synthétiques que créent les spécialistes de mesures.

Après une revue des modèles de comportement de programme les plus fréquemment utilisés, nous étudierons divers algorithmes de gestion d'une mémoire paginée à deux niveaux, ainsi que la taille de l'ensemble de travail. Des expériences menées sur des programmes soumis au Centre Interuniversitaire de Calcul de Bretagne nous ont fourni des valeurs des paramètres des modèles. Pour parvenir à une description plus fidèle de la réalité, nous proposons un nouveau modèle de comportement. Ce travail s'achève par l'étude d'algorithmes de gestion des disques qui constituent le niveau lent de la plupart des hiérarchies de mémoire.

TABLE DES MATIERES

<u>CHAPITRE PREMIER</u> :PRESENTATION DES MODELES DE COMPORTEMENT DE PROGRAMME	1
1.Les données expérimentales	2
2.Modèles à temps discret du comportement des programmes	9
3.La fonction de durée de vie	17
Références	20
<u>CHAPITRE DEUX</u> : ALGORITHMES DE REMPLACEMENT DE PAGES	23
1.Formalisation de la notion d'algorithme de remplacement de pages	26
2.Evaluation des ARP à la demande sur le modèle de programme à références indépendantes	33
3.Les algorithmes de remplacement de pages à préchargement partiel	44
4.Evaluation des ARP sur le modèle de comportement de programme à liste LRU	54
Annexe A:démonstration du théorème 1	57
Annexe B:calcul de F(MIN)	60
Références	66
<u>CHAPITRE TROIS</u> : GESTION GLOBALE DE LA MEMOIRE CENTRALE ET D'UN DISQUE A TETES FIXES	69
1.Introduction	69
2.Etude analytique des algorithmes de gestion globale à préchargement partiel	74
3.Etude en simulation des algorithmes FIFO(ℓ) et LRU(ℓ)	97
Annexe:démonstration du lemme	116
Références	119

<u>CHAPITRE QUATRE</u> : TAILLE DE L'ENSEMBLE DE TRAVAIL	121
1.Introduction	121
2.Distribution de la taille d'un ensemble de travail	124
2.1.Programmes conformes au modèle à liste LRU	124
2.2.Programmes conformes au modèle à références indépendantes	127
2.3.Résultats numériques	134
3.Références	139
 <u>CHAPITRE CINQ</u> : RESULTATS EXPERIMENTAUX	 141
1.L'outil de mesure	143
2.L'échantillon de programmes	149
3.Utilisation des instructions	152
4.Les ruptures de séquence	159
5.Propriétés des chaînes de références	161
6.Taux de défauts de page	166
7.Taille de l'ensemble de travail	174
Références	178
Annexe:regroupement des instructions du calculateur C.I.I. 10070	181
 <u>CHAPITRE SIX</u> : LE MODELE A RESEAU DE RETARD DU COMPORTEMENT DES PROGRAMMES	 183
1.Description du modèle à réseau de retards	184
2.Résultats fondamentaux sur le modèle à réseau de retards	192
3.L'ensemble de travail	204
4.L'algorithme LRU de remplacement de pages	208
5.Exemples numériques	210
6.Une extension du modèle	217
Annexe A:démonstration du théorème 1	224
Annexe B:démonstration du théorème 4	235
Références	238

<u>CHAPITRE SEPT</u> : GESTION DES MEMOIRES A DISQUE OU TAMBOUR	239
1.Introduction	239
1.1.Disques à têtes fixes et tambour	241
1.2.Disques à têtes mobiles	245
1.3.Le problème des groupes de disques	250
2.Utilisation des disques à têtes fixes	253
2.1.Analyse de la file d'attente M/G/1	253
2.2.Analyse de la politique d'Eschenbach	268
2.3.Gestion des disques à têtes fixes sans secteurs	282
3.L'utilisation des disques à têtes mobiles	296
3.1.Ordonnancement des requêtes	296
3.2.Comparaison des politiques d'ordonnancement	298
Références	312

CHAPITRE PREMIER :

PRESENTATION DES MODELES DE COMPORTEMENT DE PROGRAMME

Si vaste que soit la mémoire rapide d'un ordinateur, elle s'est toujours révélée insuffisante pour contenir l'espace d'adresse dont les programmeurs souhaitaient disposer dans mainte application. L'accroissement de la taille de ces mémoires est limité par des contingences économiques, voire même par les possibilités technologiques. Or l'information n'a besoin de se trouver en mémoire centrale (ou, le cas échéant, dans l'antémémoire) qu'au moment d'être traitée par des processeurs ; sinon elle peut être rangée dans des mémoires auxiliaires à accès plus lent mais moins coûteuses. Dans de nombreux systèmes d'exploitation récents - ceux qui utilisent le concept de mémoire virtuelle - c'est le système lui-même qui contrôle la répartition de l'information entre les différents supports disponibles et qui décide de l'opportunité des transferts. Pour que ces fonctions soient exercées dans de bonnes conditions, deux types de mécanismes ont été conçus : la segmentation de programmes ou leur pagination.

Sur quelques ordinateurs (en particulier sur des machines Burroughs) la mémoire rapide est allouée par segments. La partition en segments est fondée sur la logique du programme : un segment est une partie de l'espace d'adresse égale à une procédure, à un bloc, à un fichier ou un sous-fichier ... A l'intérieur d'un segment, chaque emplacement est désigné par rapport à une adresse origine : lors de l'exécution le support d'un segment est un bloc de mémoire centrale dont le premier emplacement est désigné par le contenu d'un registre spécial. La règle des 50 % de Knuth [1], complétée par celle des 2/3 de Gelenbe [2], a mis en valeur l'inconvénient majeur de cette allocation de la mémoire par blocs d'un seul tenant, de tailles différentes : sous certaines conditions d'équilibre, un tiers environ de la mémoire rapide est inutilisée.

C'est pour cette raison que la plupart des systèmes à mémoire virtuelle sont dotés d'un autre mécanisme, la pagination. Dans cette technique, l'espace d'adresse de chaque programme est divisé en blocs de même taille, les pages : cette partition ne reflète évidemment pas la structure logique des programmes. La mémoire centrale est elle-même partagée en "pages physiques" dont chacune peut être le support d'une page d'un programme. Un dispositif câblé, la topographie de mémoire, permet de traduire toute adresse virtuelle en une adresse physique. Notons que dans certains calculateurs (HONEYWELL 6180) la mémoire est attribuée par pages aux segments constituant chaque programme. La segmentation est alors un moyen élégant de résoudre le problème des liens ou celui de la protection. Du point de vue de l'allocation de mémoire, qui est le nôtre, ces systèmes sont essentiellement des systèmes paginés.

1 - LES DONNEES EXPERIMENTALES

La conception des mécanismes de mémoire virtuelle et l'essor de la multiprogrammation justifient l'intérêt porté au comportement des programmes et en particulier à la dispersion des références qu'ils exécutent dans l'espace d'adresse. Très tôt, il est apparu qu'un programme peut s'exécuter dans une zone de mémoire physique très inférieure à son espace d'adresse sans qu'il en résulte pour autant une dégradation notable du temps de réponse ou du taux horaire de traitement. La figure 1 est tout à fait caractéristique à cet égard. Elle représente la variation de la durée totale d'exécution d'un programme (temps de calcul + attente due aux transferts entre la mémoire centrale et la mémoire secondaire, en monoprogrammation) en fonction de la taille s de la mémoire centrale. Tant que s est supérieur à une va-

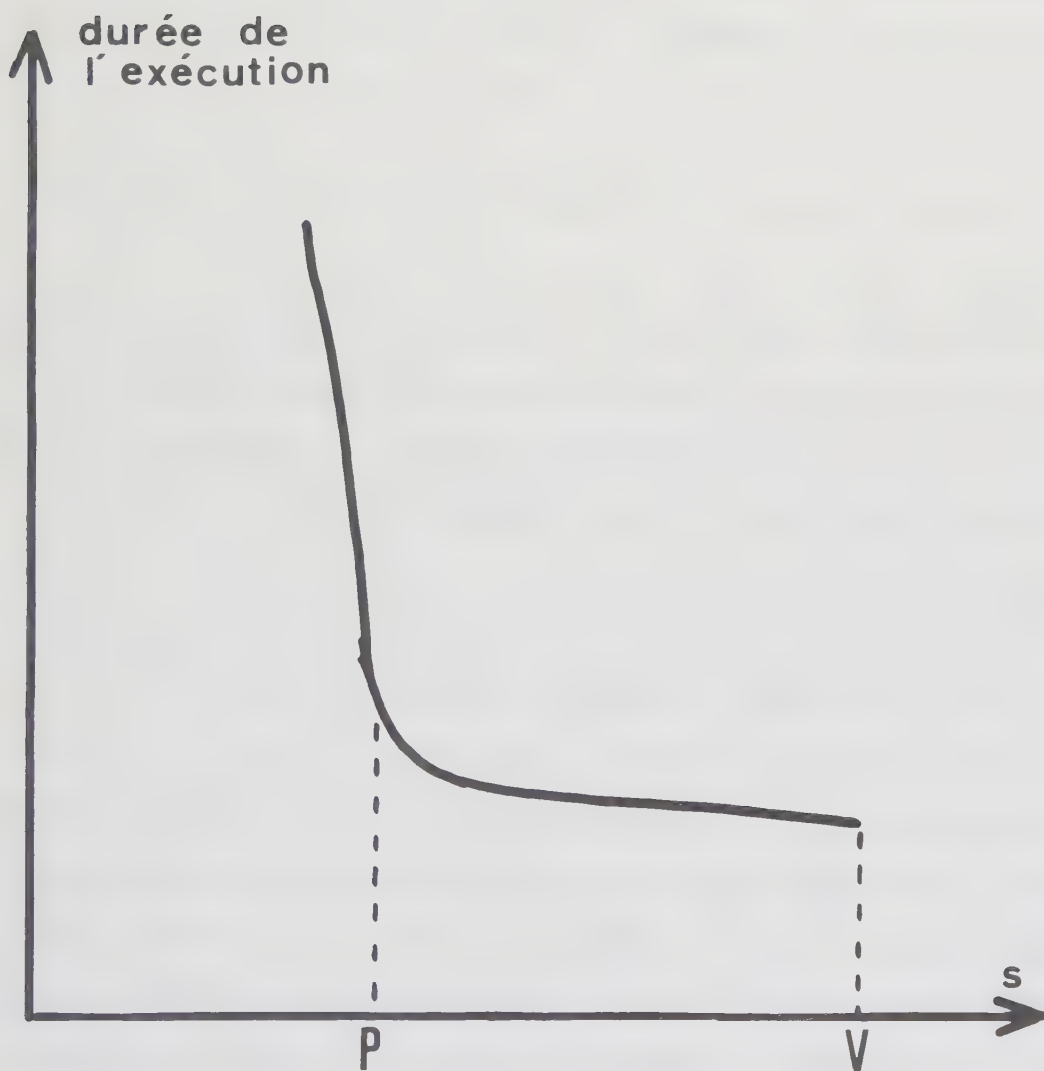


FIGURE 1

leur P la durée d'exécution est sensiblement la même que lorsque tout l'espace virtuel (taille : V) est présent en mémoire. L'exécution ne peut avoir lieu dans des conditions satisfaisantes si $s < P$: la dégradation des performances est très brutale. Le point P est parfois appelé *paracore*.

De manière informelle, les aspects principaux du comportement d'un programme sont les suivants :

= la fréquence avec laquelle un mot de mémoire (ou une page ou un segment) est référencé au cours de l'exécution révèle de profondes disparités entre les mots. Ainsi les mesures rapportées au chapitre 5 nous ont montré qu'il est fréquent que 80 % des références soient concentrées sur moins de 15 % des pages.

== l'ensemble des adresses auxquelles un programme accède durant un intervalle de temps "raisonnable" est en général beaucoup plus petit que l'espace d'adresse et il ne contient qu'une partie des pages fréquemment référencées : cet ensemble est la "localité" instantanée. En d'autres termes la probabilité qu'une adresse donnée soit référencée à l'instant t dépend pour une large part des références exécutées dans le passé récent : en particulier, elle dépend de la date de la dernière référence à cette page.

Pour formaliser cette propriété, P. J. Denning [3] a introduit le concept de travail (E.T.) : l'E.T. $W(t, T)$ est l'ensemble des adresses référencées entre $t - T$ et t . En vertu de la propriété de localité on peut s'attendre à ce que l'E.T. évolue lentement pendant presque toute la durée de l'exécution. D'autres variations, beaucoup plus brutales, sont dues, par exemple à des changements de phase.

L'existence des localités a plusieurs causes. D'une part l'attention du programmeur se porte successivement sur des fractions de l'algorithme qu'il met en oeuvre, certaines structures de données pouvant aussi favoriser cette concentration. D'autre part un programme comporte de nombreuses boucles qui, étant exécutées un grand nombre de fois, ont pour effet de circonscrire momentanément la partie utilisée de l'espace virtuel.

=== certains motifs apparaissent fréquemment dans la chaîne des pages (ou segments) successivement référencées. Si la page q est souvent référencée après la page p, cela peut avoir plusieurs causes : par exemple une boucle d'exécution à cheval sur p et q ou un appel dans p à un sous-programme se trouvant dans q. C'est sur la détection de tels motifs que reposent les techniques de restructuration des programmes dont l'objectif est d'accroître le degré de localité des programmes [4] [5].

Pour un algorithme à mettre en oeuvre et un même jeu de données, la dispersion des références exécutées dans l'espace d'adresse dépend considérablement du style de programmation et de l'organisation des données. Ce point est fort bien illustré par les travaux de Brawn et Gustavson [6]. Trois versions d'un programme d'inversion de matrice utilisant la même méthode de Gauss ont été exécutées en monoprogrammation sur l'ordinateur prototype I B M M 44/ 44 X qui est doté d'un mécanisme de pagination ; la durée totale de l'inversion d'une même matrice d'ordre 100 a été mesurée pour différentes allocations de mémoire centrale (fig. 2). Dans la version V 1, écrite en FORTRAN, la matrice à inverser est rangée par colonnes dans un tableau carré d'ordre 150 ; la lecture des données et l'impression de l'inverse se font ligne par ligne. Dans la version V 2 le tableau réservé est

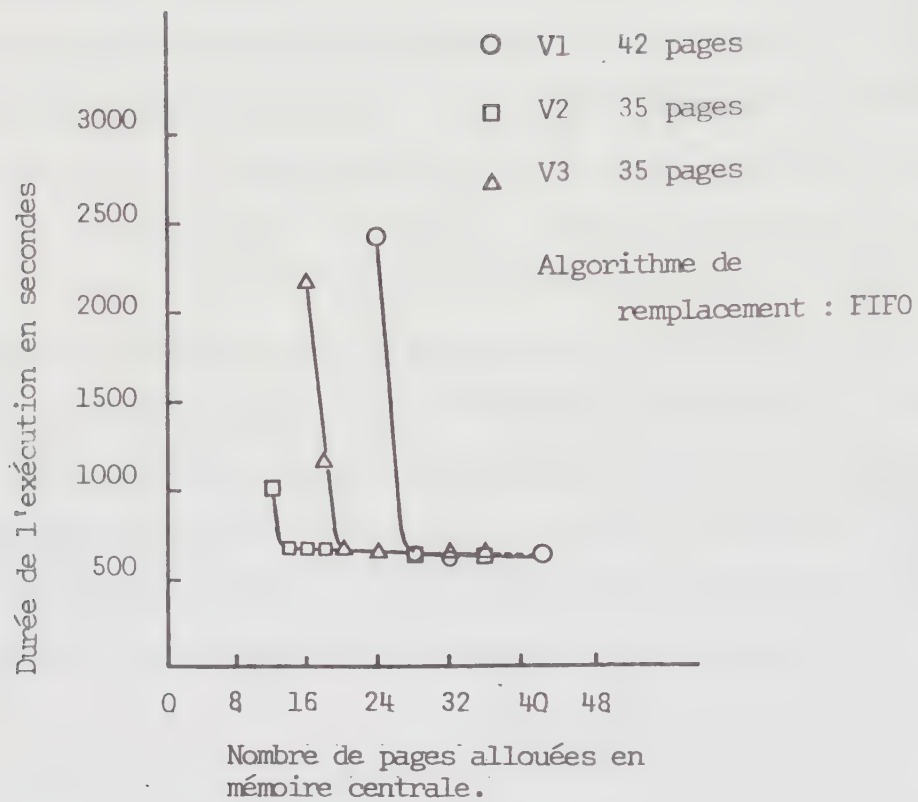


Figure 2

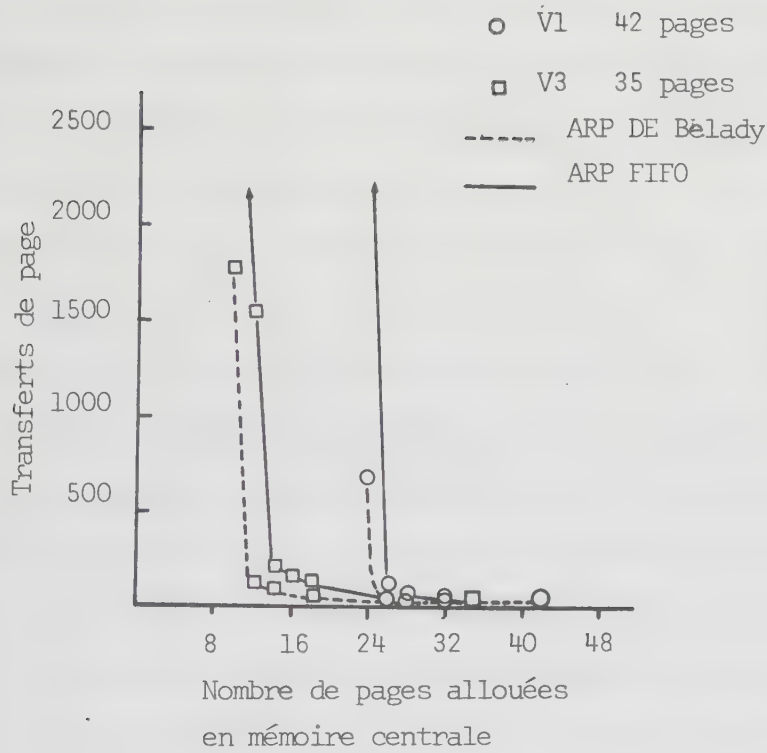


Figure 3

d'ordre 100. Il en est de même dans la version V 3 où, en outre, lecture et impression se font colonne par colonne. L'amélioration du temps de réponse du système traduit une moindre dispersion des références. En particulier elle n'a pas pour cause le comportement de l'algorithme de remplacement de pages : l'algorithme FIFO habituellement utilisé sur le système M 44/44 X ayant été remplacé par l'algorithme optimal de Belady, les mesures ont été refaites pour les versions V 1 et V 3. La figure 3 montre que le même gain est obtenu par utilisation de la version V 3 à la place de V 1.

Si l'on se livre à une étude expérimentale plus fine du comportement des programmes, on est amené à manipuler d'énormes quantités de données. Ainsi les suites d'adresses de mots référencés que nous avons relevées au cours d'exécutions interprétatives (voir chapitre 5) comportent plusieurs millions ou dizaines de millions d'éléments prenant leur valeur dans un espace dont la taille est de quelques dizaines de milliers d'adresses. Un regroupement de ces adresses est nécessaire pour que l'exploitation des mesures soient possible. Il est d'autant plus justifié que la finalité de tels travaux est d'apporter une aide à la conception des systèmes : or l'allocation de mémoire ou les échanges entre les différents niveaux d'une hiérarchie de mémoire ne prennent pas en compte des mots isolés mais des pages ou des segments.

Comme la plupart des auteurs, nous avons choisi de considérer des partitions de l'espace d'adresse en pages (de même taille) plutôt qu'en segments. Ce choix est sans doute celui qui rend les mesures les plus aisées. C'est aussi dans un espace paginé que la connaissance du comportement des programmes est la plus riche de conséquences pratiques. La première raison

en est que les systèmes à mémoire paginée sont considérablement plus nombreux que les systèmes où la mémoire physique est allouée par blocs. En outre, alors que certaines grandeurs (l'ensemble de travail, par exemple) peuvent être analysées dans un espace segmenté aussi bien que dans un espace paginé, il existe des problèmes propres à la pagination dont l'importance est très grande : conception d'algorithmes de remplacement de page, choix de la taille des pages.

2 - MODELES A TEMPS DISCRET DU COMPORTEMENT DES PROGRAMMES

Nous appellerons espace d'adresse d'un programme l'ensemble N de ses n pages virtuelles :

$$N = \{1, 2, \dots, n\}$$

La suite

$$r_0 \ r_1 \ r_2 \ \dots \ r_t \ \dots$$

des adresses auxquelles le programme accède au cours d'une exécution est considérée comme une réalisation d'un processus stochastique à temps discret $\left(R_t \right)_{t \in \mathbb{N}}$; la valeur de la variable aléatoire R_t est le numéro de la page virtuelle référencée à l'instant t : c'est un élément de N . L'unité de temps est l'intervalle moyen entre deux références consécutives à la mémoire. Remarquons qu'une réalisation de ce processus stochastique est une suite infinie alors qu'un programme ne peut évidemment exécuter qu'un nombre fini de références : cette différence n'a pas de conséquence fâcheuse, en particulier parce que nous nous limiterons à l'étude de comportements stationnaires.

Avec ce formalisme, définir un modèle de comportement c'est énoncer un ensemble d'hypothèses sur le processus $\left\{ R_t \right\}$. Dans ce paragraphe nous allons passer en revue les modèles les plus fréquemment utilisés.

2.1. LE MODELE DE DENNING ET SCHWARTZ

Pour toutes pages i et j et tous entiers strictement positifs t et x , nous désignerons par $g_{ij}(t,x)$ la probabilité que la première référence à la page j après l'instant t ait lieu à l'instant $t + x$, sachant que la page i est référencée à l'instant t :

$$g_{ij}(t,x) = \Pr \left\{ r_{t+1} \neq j ; \dots ; r_{t+x-1} \neq j ; r_{t+x} = j \mid r_t = i \right\}$$

Le modèle de Denning et Schwartz est caractérisé par les hypothèses suivantes :

D S 1 Pour tout i appartenant à N , $g_{ii}(t,x)$ ne dépend pas de t .
Nous noterons $g_{ii}(t,x) = f_i(x)$

D S 2 Pour tout i , $\sum_{x \geq 1} f_i(x) = 1$

D S 3 Pour tout i , la durée moyenne de l'intervalle X_i entre deux références successives à la page i est finie :

$$\bar{X}_i = \sum_{x \geq 1} x f_i(x) < + \infty$$

L'hypothèse la plus forte est D S 1 : elle exprime que le comportement d'un programme à l'intérieur d'une localité est homogène dans le temps. La taille moyenne de l'ensemble de travail et le taux d'absences de page

(c'est-à-dire la probabilité que r_{t+1} n'appartienne pas à $W(t, T)$) ont été calculés en fonction des f_i ; en outre une relation entre ces deux fonctions a été démontrée [7]. Ces résultats sont largement applicables puisque la plupart des modèles de comportement de programme - notamment ceux qui sont présentés dans ce paragraphe - sont des cas particuliers de celui de Denning et Schwartz. Malheureusement, à cause même de cette généralité, il ne peut être utilisé pour analyser des algorithmes de remplacement de page.

2.2. LE MODELE A REFERENCES INDEPENDANTES

Le modèle de comportement de programme est dit "à références indépendantes" si le processus stochastique $\left(R_t \right)$ satisfait aux trois conditions suivantes :

R I 1 Pour toute famille $t_1, t_2, \dots, t_p$ d'entiers positifs deux à deux distincts, les variables aléatoires $R_{t_1}, R_{t_2}, \dots, R_{t_p}$ sont indépendantes.

R I 2 Toutes les variables aléatoires R_t ont la même distribution. Pour toute page i et tout entier t , nous noterons :

$$\Pr \left(R_t = i \right) = \beta_i$$

R I 3 Pour tout i , β_i est différent de 0

Remarquons que ce modèle ne prend pas en compte les propriétés de localité des programmes. En revanche il traduit l'importance relative - en durée d'exécution - des différentes parties du programme. Statistique-

ment la probabilité β_i apparaît comme la fraction des références qui, au cours d'une longue période, concernent la page i .

Ce modèle de comportement est un cas particulier de celui qui est défini en 1. 2. 3. En effet, l'intervalle entre deux références successives à la page i a une distribution indépendante de t :

$$f_i(x) = \left(1 - \beta_i\right)^{x-1} \beta_i \quad (x \geq 1)$$

qui vérifie en outre

$$\sum_{x \geq 1} f_i(x) = \beta_i \sum_{x \geq 1} \left(1 - \beta_i\right)^{x-1} = \frac{\beta_i}{\beta_i} = 1$$

et

$$\sum_{x \geq 1} x f_i(x) = \beta_i \sum_{x \geq 1} x \left(1 - \beta_i\right)^{x-1} = 1 / \beta_i < +\infty$$

Notons que si le comportement d'un programme dans son espace d'adresse est à références indépendantes, il en est de même lorsque la taille des pages est doublée : la page obtenue par réunion des pages i et j est référencée avec probabilité $\beta_i + \beta_j$. Cette propriété de stabilité du modèle à références indépendantes est exceptionnelle si l'on considère les différents modèles proposés. Elle ne s'applique d'ailleurs pas à la division par deux de la taille de page.

Nous utiliserons ce modèle pour l'analyse de divers algorithmes de remplacement de page ainsi que pour l'étude de l'ensemble de travail.

2.3. LE MODELE MARKOVIEN D'ORDRE r

Soit r un entier non négatif. Le processus stochastique $\left(R_t\right)$ est

markovien d'ordre r si et seulement si pour tout entier t plus grand que $r - 1$, pour toute famille $\left\{ a_i \right\}_{i=0, 1, \dots, r}$ de $(r + 1)$ pages, pour tout ensemble fini $\left\{ t_i \right\}_{i \in I}$ d'entiers non négatifs et inférieurs à $t - r$, pour toute famille $\left\{ b_i \right\}_{i \in I}$ de pages, l'égalité suivante est vérifiée :

$$\Pr \left(R_t = a_0 \mid (\forall i) (1 \leq i \leq r) : R_{t-i} = a_i \right) =$$

$$\Pr \left(R_t = a_0 \mid (\forall i) (1 \leq i \leq r) : R_{t-i} = a_i ; (\forall i \in I) : R_{t_i} = b_i \right)$$

Dans ce modèle les pages sont référencées avec des fréquences différentes. En ce qui concerne la localité des programmes, l'influence du passé est réduite aux r dernières références. Les études se limitent au cas où les probabilités conditionnées

$$\Pr \left(R_t = a_0 \mid (\forall i) (1 \leq i \leq r) : R_{t-i} = a_i \right)$$

sont indépendantes du temps. Le modèle markovien d'ordre 0 est alors le modèle à références indépendantes sous réserve que la condition R I 3 soit vérifiée.

Pour étudier un tel comportement on peut associer au processus $\left\{ R_t \right\}$ la chaîne de Markov $\mathcal{C}$ (d'ordre 1) dont l'ensemble des états est N^r . L'état $\left\{ a_1, a_2, \dots, a_r \right\}$ de cette chaîne à l'instant t est le r -uple des références exécutées aux instants $t - 1, t - 2, \dots, t - r$. Les seules transitions dont la probabilité n'est pas a priori nulle sont celles d'un état quelconque, noté $\left\{ a_1, a_2, \dots, a_r \right\}$, à un état de la forme $\left\{ a, a_1, a_2, \dots, a_{r-1} \right\}$. Ces transitions sont au nombre de n^{r+1} . Pour comparer un programme réel ayant vingt pages virtuelles à son modèle markovien d'ordre 5, il serait nécessaire de mesurer $20^6 = 64\,000\,000$ probabilités. Aussi les

modèles markoviens ne sont utilisés que si leur ordre est 0 ou 1 [8]. Ceci réduit considérablement leur aptitude à représenter le comportement local des programmes.

2.4. LES MODELES A LISTE L R U

Ces modèles ont été conçus pour prendre en compte les bonnes performances de l'algorithme de remplacement de pages L R U. Ils déterminent la probabilité qu'une page donnée soit la prochaine à être référencée par la considération de sa position dans la liste L R U c'est-à-dire dans la liste des pages ordonnées selon la date de leur dernière utilisation.

L'ensemble des états du programme est l'ensemble des permutations sur N. Une permutation est notée $\sigma = \left[a_1, a_2, \dots, a_k, \dots, a_n \right]$ et l'on dit que la page a_k est à la distance k dans σ . Un état initial σ_0 étant choisi, à toute suite de références

$$r_0 \ r_1 \ r_2 \quad \dots \ r_t \ r_{t+1} \quad \dots$$

est associée la suite d'états du programme

$$\sigma_0 \ \sigma_1 \ \sigma_2 \quad \dots \ \sigma_t \ \sigma_{t+1} \quad \dots$$

définie par la relation de récurrence suivante :

Si l'on note $\sigma_t = \left[a_1, a_2, \dots, a_n \right]$ et si la page r_t est à la distance d_t dans σ_t (c'est-à-dire que $a_{d_t} = r_t$) alors :

$$\sigma_{t+1} = \left[a_{d_t}, a_1, a_2, \dots, a_{d_t-1}, a_{d_t+1}, \dots, a_n \right]$$

La connaissance de la suite $\left[r_t \right]$ équivaut à celle de $\left[d_t \right]$: dans les modèles à liste L R U, $\left[d_t \right]$ est considérée comme la réalisation d'un processus stochastique $\left[D_t \right]_{t \in \mathbb{N}}$

Shedler et Tung [9] ont proposé de considérer $\left(D_t \right)_{t \in \mathbb{N}}$ comme une chaîne de Markov. Dans [10] et [11], les variables aléatoires D_t sont considérées comme indépendantes et identiquement distribuées. Nous nous limiterons désormais à cette hypothèse. Le modèle de comportement de programme est donc "à liste L R U " si le processus stochastique $\left(R_t \right)_{t \in \mathbb{N}}$ est lié à un processus $\left(D_t \right)_{t \in \mathbb{N}}$ satisfaisant aux trois hypothèses suivantes :

L R U 1 Pour toute partie finie $\{t_1, t_2, \dots, t_p\}$ de $\mathbb{N}$, les variables aléatoires $D_{t_1}, D_{t_2}, \dots, D_{t_p}$ sont indépendantes.

L R U 2 Toutes les variables aléatoires D_t ont la même distribution. Pour tout $1 \leq i \leq n$, nous poserons $\Pr \left(D_t = i \right) = \alpha_i$

L R U 3 Pour tout i , α_i est strictement positif.

Si ces trois conditions sont vérifiées, la suite $\left(\sigma_t \right)$ des états du programme est la réalisation d'une chaîne de Markov régulière. La matrice de transition de cette chaîne étant bistochastique tous les états ont la même probabilité stationnaire : $\frac{1}{n!}$. Lorsqu'un programme à comportement L R U est en équilibre statistique, toutes ses pages sont référencées avec la même fréquence ; l'intervalle moyen entre deux références successives à une même page est n .

Ce modèle est lui aussi un cas particulier du modèle de Denning et Schwartz. Calculons la distribution de l'intervalle X_i entre deux références successives à la page i . Une page reste à la distance j ($2 \leq j \leq n$) dans la liste L R U tant que les pages référencées sont des pages se trou-

vant à une distance au plus égale à $j - 1$; la probabilité que ce séjour dure exactement y unités de temps est donc

$$(1) \quad S_j^y = \frac{1}{1 - S_j} \left(1 - S_j - 1 \right)$$

où l'on a posé

$$S_k = \sum_{r=1}^k \alpha_r \quad (1 \leq k \leq n).$$

La fonction génératrice de la distribution de ce temps de séjour est

$$(2) \quad H_j(z) = \sum_{y=1}^{\infty} \left(1 - S_j - 1 \right) S_j^{y-1} z^y = \frac{\left(1 - S_j - 1 \right)}{1 - S_j - 1} \frac{1}{z}$$

Lorsqu'une page quitte la distance j ($2 \leq j \leq n$), elle passe à la distance $j + 1$ (probabilité : $\left(1 - S_j \right) / \left(1 - S_j - 1 \right)$) ou à la distance 1, c'est-à-dire qu'elle a été référencée (probabilité : $\alpha_j / \left(1 - S_j - 1 \right)$).

La probabilité que la distance maximum atteinte par la page i entre deux références soit k ($2 \leq k \leq n$) est donc égale à :

$$\left(1 - S_1 \right) \frac{1 - S_2}{1 - S_1} \frac{1 - S_3}{1 - S_2} \cdots \frac{1 - S_{k-1}}{1 - S_{k-2}} \frac{\alpha_k}{1 - S_{k-1}}$$

c'est-à-dire à α_k . La durée X_i étant inférieure de une unité à la somme des temps passés par la page i aux distances 2, 3, ..., k , on en déduit l'expression suivante de la fonction génératrice de la distribution de X_i :

$$F_i^*(z) = \sum_{x \geq 1} f_i(x) z^x = \alpha_1 z + \sum_{k=2}^n \alpha_k z \prod_{j=2}^k H_j(z)$$

soit :

$$F_i^*(z) = \sum_{k=1}^n \alpha_k z \prod_{j=2}^k \frac{(1 - S_j - 1) z}{1 - S_j - 1 z}$$

Il est aisé de montrer que $F_i^*(1) = 1$ et que $E \left(X_i \right) = \lim_{z \rightarrow 1} \frac{d F_i^*(z)}{dz} = n$, ce qui constitue une vérification des conditions D S 2 et D S 3.

Ce modèle de comportement de programme est utilisé dans la simulation de la gestion d'une mémoire hiérarchisée que nous présentons au chapitre 3 . Il sert à l'analyse de l'ensemble de travail (chapitre 4). Son aptitude à représenter le comportement de programmes réels est évaluée au chapitre 5 .

3 - LA FONCTION DE DUREE DE VIE

Les modèles que nous avons présentés ci-dessus tendent à représenter le comportement des programmes à une échelle "microscopique". Ils permettent d'évaluer les algorithmes de remplacement de page et certaines règles d'allocation de la mémoire. Par leur complexité ou par leur niveau de détail, ils ne sont pas adaptés à certaines études. Ainsi leur emploi se révèle très délicat dans la simulation d'une hiérarchie de mémoire à trois niveaux :

antémémoire - mémoire centrale - mémoire secondaire

Tout en restant dans une gamme de paramètres raisonnables il peut être nécessaire de simuler une centaine de milliers de références - en moyenne - pour observer un défaut entre la mémoire centrale et la mémoire secondaire. L'obtention de résultats statistiquement significatifs peut se révéler très coûteuse dans ces conditions.

La fonction de durée de vie est un modèle très maniable du comportement des programmes. La durée de vie $e(s)$ est le temps de calcul moyen en-

tre deux références hors de la mémoire rapide pour un programme auquel est allouée une zone de taille s dans cette mémoire rapide. La forme générale de la courbe $e(s)$ dépend assez peu du programme ou de la politique selon laquelle la mémoire est gérée. Après avoir observé l'exécution de programmes dans des zones de mémoire de tailles différentes, Belady et Kuehner [12] ont suggéré une approximation de $e(s)$ valable tant que s est inférieur à une valeur s_0 qui est voisine du paracore :

$$e(s) \approx a s^k$$

Dans cette formule, la vitesse d'exécution du processeur et l'influence de l'algorithme de remplacement de page se reflètent dans le coefficient a . L'exposant k traduit la plus ou moins grande dispersion des références exécutées par le programme : c'est le degré de localité, dont la valeur est le plus souvent comprise entre 1,5 et 2,5. Une courbe mesurée et son approximation sont représentées sur la figure 4. La partie de la courbe concernée par l'approximation de Belady et Kuehner est celle où la variation de la durée de vie est la plus rapide : cette approximation peut donc suffire à l'étude de politiques optimales d'allocation de la mémoire. Néanmoins, Chamberlin, Fuller et Liu [13] ont proposé une autre approximation qui prend en compte toute la courbe :

$$e^*(s) = \frac{2B}{1 + \left(\frac{C}{s}\right)^2}$$

La fonction de durée de vie a été employée par Gelenbe pour représenter le comportement des programmes dans une hiérarchie de mémoire à trois niveaux [14]. Elle a été introduite également dans des modèles analytiques [15] ou des modèles de simulation [13] conçus pour l'étude de la multiprogrammation.

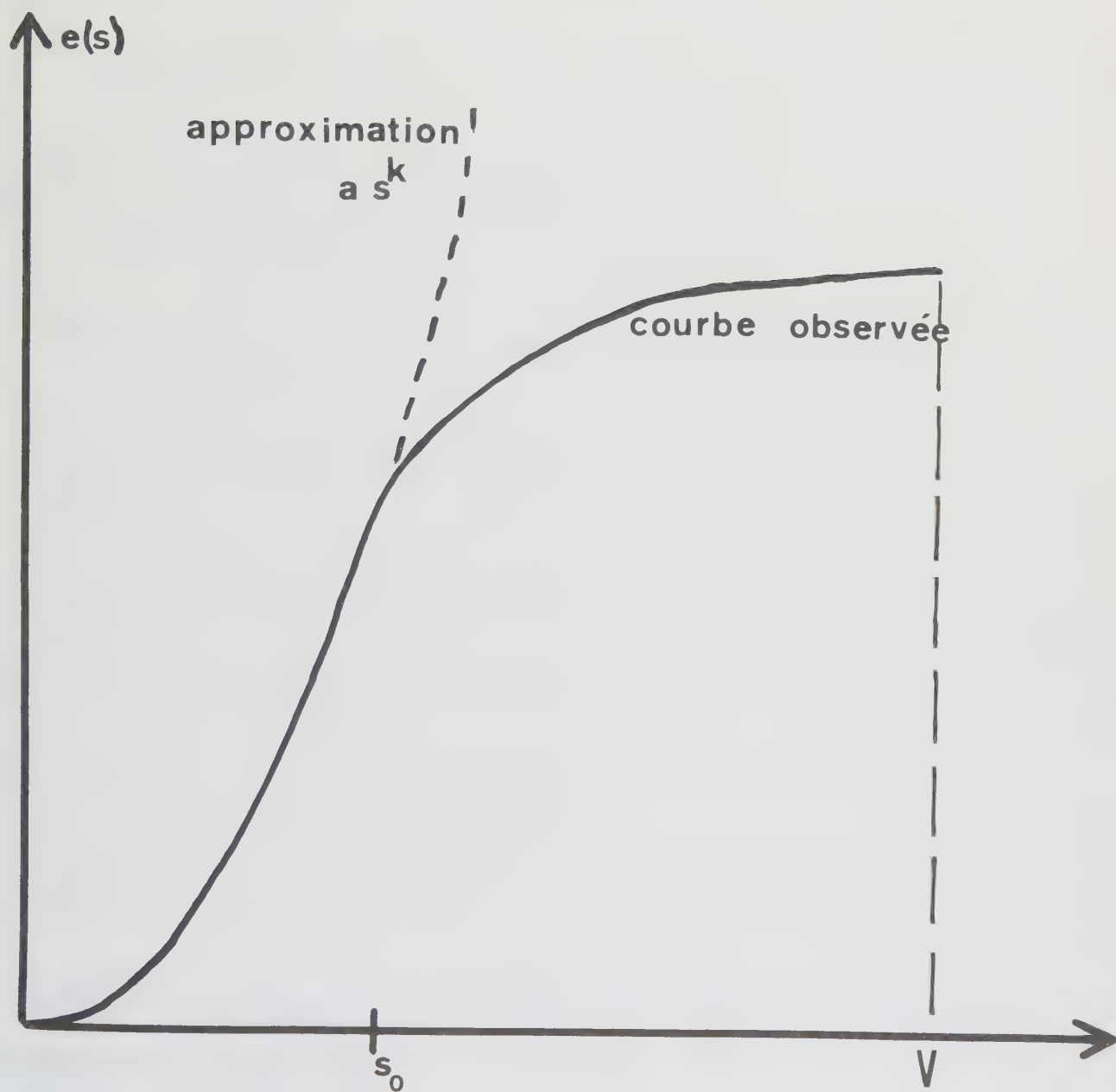


FIGURE 4

REFERENCES

- [1] KNUTH D. : The art of programming : I, pp 445-446, Addison-Wesley, Reading, Mass. (1968)
- [2] GELENBE E. : The two thirds rule for dynamic storage allocation under equilibrium - Information Processing letters 1, 59-60 (1971)
- [3] DENNING P.J. : The working set model for program behavior - Comm. A C M 11, 323-333 (1968)
- [4] HATFIELD D.J. - GERALD J. : Program restructuring for virtual memory - I B M Syst. J. 10, 168-192 (1971)
- [5] FERRARI D : The method of critical working set for the automatic improvement of program locality - Tech. Rep. University of California, Berkeley
- [6] BRAWN B.S. - GUSTAVSON F.G. : Program behavior in a paging environment - Proc. AFIPS FJCC 1968, pp 1019-1032
- [7] DENNING P.J. - SCHWARTZ S.C. : Properties of the working set model - Comm. A C M 15, 191-198 (1972)
- [8] FRANKLIN M.A. - GUPTA R.K. : Computation of page fault probability from program transition diagram - Comm. A C M 17, 168-191 (1974)
- [9] SHEDLER G.S. - TUNG C. : Locality in page reference strings - SIAM J. on Comp. 1, 218-241 (1972)
- [10] DENNING P.J. - SPIRN J.R. - SAVAGE J.E. : Some thoughts about locality in program behavior - Proc. Symposium on computer communication networks and teletraffic, Polytechnic Institute of Brooklyn (1972), pp 101-112
- [11] LENFANT J. : Evaluation sur des modèles de comportement de la taille d'un ensemble de travail - RAIRO Revue Bleue, B2 (1974)

- [12] BELADY L.A., KUEHNER C.J. : Dynamic space sharing in computer systems - Comm ACM 12,282-288 (1969)
- [13] CHAMBERLIN D.D., FULLER S.H., LIU L.Y. : An analysis of page allocation strategies for multiprogramming systems with virtual memory - IBM J. Res. Develop 17,404-412 (1973)
- [14] GELENBE E. : The distribution of a program in primary and fast buffer storage - Comm. ACM 16,431-434 (1973)
- [15] BADEL M., GELENBE E., LENFANT J., LEROUDIER J., POTIER D. : Adaptive optimization of the performance of a virtual memory computer - Proc. SIGMETRICS Conference (1974)

CHAPITRE DEUX :

ALGORITHMES DE REMPLACEMENT DE PAGES

L'organisation des transferts entre le niveau de mémoire rapide et le niveau de mémoire lente est un des principaux problèmes que soulève la gestion des hiérarchies de mémoire. Les notions de mémoire lente et de mémoire rapide sont d'ailleurs toutes relatives. Dans les grands systèmes récents, le problème se pose pour chacun des couples de deux niveaux successifs. Les cas typiques sont :

mémoire centrale à tores ou circuits intégrés - mémoire secondaire
à disques ou tambours
antémémoire ("cache") - mémoire centrale
mémoire centrale - mémoire de migration (Large Capacity Storage,
Extended Core Storage, mémoire à bulles)
mémoire de migration - mémoire secondaire.

L'information ne doit se trouver au niveau rapide qu'au moment d'être traitée ; sinon elle peut séjourner dans le niveau lent. La réalisation d'un système à plusieurs niveaux de mémoire paginée nécessite des choix en réponse aux questions suivantes :

- dans quelle page physique du niveau rapide charger une page dont le transfert est apparu nécessaire ?
- à quel instant lancer un transfert ?
- quelles pages charger en mémoire ?
- quelle page physique libérer en mémoire rapide pour fournir une place à une page dont la présence y est devenue nécessaire ?

La première question à résoudre n'implique pas, en général, de choix significatif : les registres topographiques par l'intermédiaire desquels sont traduites les adresses virtuelles peuvent pointer sur n'importe quelle page physique. Le cas du calculateur I B M 360/195 est exceptionnel [1]. Dans cette machine la mémoire centrale et l'antémémoire sont organisées en matrices de pages ayant le même nombre de colonnes. Un transfert entre ces deux niveaux n'est possible qu'entre colonnes de même rang ; lors d'un chargement à partir de la mémoire secondaire, le choix de l'emplacement en mémoire principale n'est donc pas indifférent.

Selon la réponse apportée à la seconde question, on classe les politiques de gestion des mémoires hiérarchisées en deux grandes catégories. Le cas le plus fréquent est celui des politiques "à la demande" : les ordres de transfert ne sont émis que lorsqu'une page référencée ne se trouve pas dans la mémoire rapide. Lorsqu'un tel évènement, appelé "défaut de page", survient, la page ayant fait défaut - et elle seule - est chargée en mémoire rapide. Si aucune page physique n'est inoccupée à ce niveau, un algorithme, dit "de remplacement de page", est activé pour déterminer la page qui cédera sa place. Le cas échéant la page remplacée devra être recopiée dans la mémoire lente avant que le chargement de la page ayant fait défaut puisse avoir lieu.

Si la mémoire lente est une mémoire rotative, les possibilités d'optimisation sont telles que le transfert de k pages dure moins longtemps que k transferts d'une page. En outre tout défaut de page provoque un appel au moniteur, ce qui se solde au minimum par des coûts de changement de contexte. Ces deux considérations justifient l'intérêt porté aux politiques de préchargement.

Dans une telle gestion, l'entrée d'une page dans le niveau rapide peut être déclenchée avant que la nécessité en soit apparue. Pratiquement cela est souvent réalisé par lancement de plusieurs transferts à l'occasion d'un même défaut de page. La grande difficulté de la mise en oeuvre de telles politiques réside dans la prédiction de l'utilité des pages à court terme, le danger étant de trop précharger de pages qui ne seront pas référencées [2]. Par ailleurs la nécessité d'un algorithme de remplacement subsiste.

De nombreuses variations autour de ces deux grandes catégories ont été imaginées. Un préchargement initial de plusieurs pages est souvent exécuté, même si la mémoire est ensuite gérée à la demande. Dans le cadre du projet ESOPÉ [3] a été testée une procédure de prévidage. Dans ce système, une page découplée est inutile, au moins pendant un certain temps : le moniteur peut donc profiter d'une période d'inactivité du canal ad hoc pour recopier cette page dans la mémoire lente. Il s'agit alors d'une anticipation sur le remplacement. Le contenu de la mémoire rapide n'étant pas modifié, le seul risque est de retarder une demande de transfert qui survient alors qu'un prévidage est en cours. Le prévidage est donc avantageux quand la charge du canal est moyenne ; il est nuisible lorsque cette charge est élevée.

Dans ce chapitre nous allons étudier les politiques de gestion "à la demande". Par ailleurs nous supposerons qu'au programme considéré est allouée une zone de taille fixe (m pages) en mémoire rapide. L'objectif est donc de trouver un algorithme de remplacement de page qui minimise le taux de défauts de page tout en restant aisé à mettre en oeuvre.

Il est clair que l'influence des défauts de page sur l'efficacité du

système est particulièrement néfaste. Pendant les transferts, l'exécution du programme ayant subi un défaut de page doit être suspendue. Du point de vue de l'utilisateur la durée totale de ces suspensions est un élément primordial du temps de réponse. Le nombre de programmes traités par le calculateur dans un temps donné est lui aussi très sensible à une augmentation du taux de défauts de page, même si le système d'exploitation est multiprogrammé. Dans ce cas si le taux de défauts de page est élevé, il en est de même de la probabilité que tous ces programmes soient suspendus et donc que l'unité centrale soit inactive. En outre l'encombrement des chemins de données et les vols de cycles - mémoire affaiblissent la vitesse d'exécution du calculateur. Enfin le temps de calcul nécessaire au traitement de ces défauts de page par le moniteur peut devenir important.

1- FORMALISATION DE LA NOTION D'ALGORITHME DE REMPLACEMENT DE PAGES

Considérons un programme dont l'espace d'adresse est $N = \{1, 2, \dots, n\}$ et auquel m pages ($m < n$) sont attribuées en mémoire.

1.1. DEFINITION DES ALGORITHMES DE REMPLACEMENT DE PAGES

Définition 1

Un état de mémoire est une partie à m éléments de l'espace d'adresse N . L'ensemble des états de mémoire sera noté S_m :

$$S_m = \{ s \mid s \subset N \text{ et } |s| = m \}$$

Définition 2

Un algorithme de remplacement de pages (ARP) est un système

où $B = (N, S, Q, \{M(r)\}, s_0, q_0)$

S est une partie non vide de S_m

Q est un ensemble fini non vide (états de contrôle)

$\{M(r)\}$ est une famille, indexée par N , de matrices

stochastiques d'ordre $|S| \cdot |Q|$

s_0 est un élément de S (état initial de la mémoire)

q_0 est un élément de Q (état de contrôle initial)

Chaque paire (s, q) de $S \times Q$ est appelée une configuration. A chaque configuration (s, q) , nous attacherons un entier distinct $i = v(s, q)$ de sorte que $1 \leq i \leq |S| \cdot |Q|$ et que $v(s_0, q_0) = 1$. La famille $\{M(r)\}$ peut être interprétée de la manière suivante. Pour toute page r de N , désignons par $m_{ij}(r)$ le terme général de la matrice $M(r)$. Si le programme référence la page r et que B se trouve dans la configuration (s, q) telle que $v(s, q) = i$, la probabilité que la nouvelle configuration soit (s', q') telle que $v(s', q') = j$ est égale à $m_{ij}(r)$.

Soit $R_1, R_2, \dots, R_t, \dots$ le processus stochastique dont une réalisation est la suite des pages référencées lors d'une exécution du programme considéré. En réponse à ces références, l'ARP B passe par une suite de configurations

$$Y_0 = (s_0, q_0), Y_1, Y_2, \dots, Y_t, \dots$$

telle que

$$(1) \quad Y_t \text{ ne dépend que de } Y_{t-1} \text{ et de } R_t \quad (t \geq 1)$$

$$(2) \quad \Pr[v(Y_1) = j \mid R_1 = r] = m_{1j}(r)$$

$$(3) \quad \Pr[v(Y_{t+1}) = j \mid v(Y_t) = i; R_{t+1} = r] = m_{ij}(r)$$

$$(1 \leq i, j \leq |S| \cdot |Q|, t \geq 1)$$

Exemple 1

On compare souvent les performances des algorithmes de remplacement avec celles de l'ARP aléatoire RAND. RAND est l'ARP qui, à l'occasion d'un défaut de page, élimine une page quelconque de la mémoire (chacune des m pages en mémoire a la probabilité $1/m$ d'être remplacée).

Dans notre formalisation RAND est tel que

$$Q = \{q_0\} \quad , \quad |Q| = 1$$

$$S = S_m$$

L'état initial de la mémoire est un élément quelconque de S .

L'élément $m_{ij}(r)$ de la matrice de transition $M(r)$ est défini par :

$$m_{ij}(r) = \begin{cases} \frac{1}{m} & \text{si les configurations } i = v(s, q_0) \text{ et } j = v(s', q_0) \\ & \text{sont telles que } r \text{ soit une page de } s' \text{ mais pas de } s \\ & \text{et qu'en outre il existe une page } p \text{ de } s \text{ vérifiant} \\ & \quad s' = s - \{p\} \cup \{r\} \\ \\ 1 & \text{si les configurations } i = v(s, q_0) \text{ et } j = v(s', q_0) \\ & \text{sont telles que } r \in s \text{ et } s = s' \\ \\ 0 & \text{pour les couples } (i, j) \text{ autres que ceux qui sont envi-} \\ & \text{sagés ci-dessus.} \end{cases}$$

Exemple 2

L'algorithme FIFO (First In - First Out) est souvent mis en oeuvre dans des systèmes d'exploitation (par exemple dans le système UMPS du Michigan Time-Sharing System [4] qui conduit un calculateur IBM 360/67)

pour sa simplicité et le peu de temps de calcul supplémentaire qu'il requiert. Cet ARP remplace la page qui a le plus long temps de présence ininterrompue en mémoire rapide. Il est formellement défini par :

- . $S = S_m$
- . Q est l'ensemble des m -uples $(p_1, \dots, p_i, \dots, p_j, \dots, p_m)$ tels que $p_i \in N$ ($1 \leq i \leq m$) et $p_i \neq p_j$ si $i \neq j$
- . Si i et j sont les numéros des configurations (s, q) et (s', q') , l'état de contrôle q étant le m -uplet $(p_1, \dots, p_m)$, pour toute page r le coefficient $m_{ij}(r)$ de la matrice $M(r)$ est défini par

$$m_{ij}(r) = \begin{cases} 1 & \text{si } r \in s, \quad s=s' \quad \text{et} \quad q=q' \\ 1 & \text{si } r \notin s, \quad s'=s \cup \{r\} - \{p_m\} \quad \text{et qu'en outre} \\ & q' = (r, p_1, \dots, p_{m-1}) \\ 0 & \text{sinon} \end{cases}$$

- . s_0 est un élément quelconque de S
- . q_0 est l'un des $m!$ états de contrôle dont toutes les composantes sont des éléments de s_0 .

Nous remarquons que FIFO est un ARP déterministe, et que (s, q) est une configuration possible de cet ARP si et seulement si chaque composante du m -uplet q est aussi un élément de l'ensemble s . Les éléments de q sont ordonnés de gauche à droite par ancienneté croissante de présence ininterrompue en mémoire.

Exemple 3

L'algorithme LRU (Least Recently Used) est un ARP dont les performances sont généralement supérieures à celles de FIFO. Lors d'un défaut de page il remplace celle des pages en mémoire rapide qui a été le plus anciennement référencée. Sa réalisation exige qu'une liste des pages de la mémoire rapide, classées selon la date de leur plus récente utilisation, soit remise à jour chaque fois qu'un accès à la mémoire est exécuté. Ceci exclut une mise en oeuvre logicielle ; néanmoins cet algorithme est fréquemment employé dans les gestions câblées [5] . L'ARP LRU est tel que

$$. S = S_m$$

. L'ensemble d'états de contrôle Q est le même que pour FIFO

. Si i et j sont les numéros des configurations (s,q) et (s',q'), l'état de contrôle q étant le m-uple $(p_1, p_2, \dots, p_m)$, pour toute page r le coefficient $m_{ij}(r)$ de la matrice M(r) est défini par

$$m_{ij}(r) = \begin{cases} 1 & \text{si } r \in s, s=s' \text{ et qu'il existe un indice k tel que} \\ & \quad . p_k = r \quad \text{et} \\ & \quad . q' = (r, p_1, \dots, p_{k-1}, p_{k+1}, \dots, p_m) \\ \\ 1 & \text{si } r \notin s, s'=s \cup \{r\} - \{p_m\} \quad \text{et si en outre} \\ & \quad q' = (r, p_1, p_2, \dots, p_{m-1}) \\ \\ 0 & \text{sinon} \end{cases}$$

. q_0 est un élément quelconque de Q

. s_0 est l'ensemble des composantes de q_0

Cet algorithme est également déterministe et l'ensemble des configurations possibles est le même que pour FIFO.

Définition 3

Un ARP est dit "à la demande" si pour tous $i = v(s, q)$ et $j = v(s', q')$ tels que $m_{ij}(r) = 0$, la page r est la seule qui appartienne à s' sans nécessairement appartenir à s .

1.2. LA MISE A JOUR DE LA MEMOIRE LENTE.

Lorsqu'un défaut de page se produit, le transfert vers la mémoire rapide de la page ayant fait défaut est inévitable.

En outre il peut être nécessaire, au préalable, de recopier dans la mémoire lente la page remplacée : c'est le cas, en particulier, si le contenu de cette page a été modifié depuis le précédent chargement. Dans certaines architectures (par exemple : couple antémémoire - mémoire principale) ce transfert supplémentaire est évité par une mise à jour quasi simultanée de la mémoire lente. Pour tenir compte de cet aspect important, nous adjoindrons au processus $(R_t)_{t \in \mathbb{N}}$, un processus $(D_t)_{t \in \mathbb{N}}$ dont les variables aléatoires ont la valeur 0 ou 1

- si $D_t = 1$ la t^e référence est considérée comme ayant modifié le contenu de la page R_t (accès pour écriture)

- si $D_t = 0$ la t^e référence n'a pas altéré le contenu de la page R_t (accès pour lecture de données ou recherche d'instruction).

Des mesures effectuées sur les programmes s'exécutant sur un calculateur CII 10070 ont montré que 70 à 75% des accès à la mémoire principale qui sont exécutés par l'unité centrale ont pour cause la recherche d'une instruction. Cette disparité entre instructions et données est due principalement à la fréquence des branchements (qui, à moins d'adressage indirect, ne néces-

sitent pas de recours à la mémoire centrale) et à celle des opérations entre registres généraux.⁽¹⁾ Les accès aux données se répartissent à peu près également entre accès en écriture et accès en lecture.

Certains compilateurs séparent le code des données pour les programmes qui s'exécutent dans un système à mémoire virtuelle. Ainsi le compilateur LP70 du système SAM (Système à Accès Multiples exploitant un CII 10070) génère trois sections de contrôle qui seront cadrées sur des frontières de page ; elles contiennent respectivement le code du programme utilisateur, ses données et les données de synchronisation (sémaphores, évènements...). Si le programme source est ainsi compilé, nous désignerons par N_c (resp. N_d) l'ensemble des pages contenant le code (resp. les données); $\{N_c, N_d\}$ constitue une partition de N :

$$N_c \cup N_d = N$$

$$N_c \cap N_d = \emptyset$$

Pour signaler si une page présente en mémoire rapide diffère de son image en mémoire lente, on peut adjoindre à chaque page physique i une bascule dont la valeur b_i^t à l'instant t est 1 ou 0 selon que la page virtuelle qui est alors logée dans i a été ou n'a pas été modifiée depuis son chargement. Désignons par $C_t(i)$ la page virtuelle présente dans la page i et par $Y_t = (s_t, q_t)$ la configuration de l'ARP à l'instant t . Nous pouvons définir de manière récursive la suite des b_i^t ; pour tout i ($1 \leq i \leq m$) :

$$b_i^0 = 0$$

(1) Dans ces mesures nous avons essayé de nous dégager des particularités du calculateur CII 10070. Dans une opération entre deux registres généraux, le second opérande est d'abord recherché en mémoire principale de sorte que l'exécution d'une telle instruction est plus longue si le second opérande est dans un registre général. Nous n'avons pas comptabilisé ces accès "parasites".

. pour tout entier t strictement positif :

$$b_i^t = \begin{cases} 1 & \text{si } r_t = C_t(i), d_t = 1 \text{ et } r_t \in s_t \\ 0 & \text{si } r_t \notin s \text{ et } C_t(i) \neq C_{t+1}(i) \\ b_i^{t-1} & \text{sinon} \end{cases}$$

2- EVALUATION DES ARP A LA DEMANDE SUR LE MODELE DE PROGRAMME A REFERENCES

INDEPENDANTES

Nous allons étudier l'efficacité des ARP en choisissant comme critère le taux de défauts de page, c'est-à-dire le nombre moyen de défauts de page survenant dans l'unité de temps : l'importance de cette grandeur a déjà été soulignée. Il est clair que les résultats que l'on peut obtenir quant à l'efficacité des ARP dépendent étroitement de la population de programmes qui a été considérée ; cela vaut aussi bien pour des mesures que pour une étude analytique. Dans ce paragraphe nous supposerons que le programme a un comportement conforme au modèle à références indépendantes, la page virtuelle i étant référencée avec probabilité β_i . Une conséquence essentielle de cette restriction est que le processus stochastique $(Y_t)_{t \in \mathbb{N}}$ est une chaîne de Markov (on s'en assurera en utilisant les hypothèses (1) (2) (3) du paragraphe 1.1)

2.1. TAUX DE DEFAUTS DE PAGE ET TAUX DE VIDAGE

Définition 4

Pour un ARP à la demande, une transition de la configuration (s, q) à

la configuration (s',q') est dite "transition avec défaut de page" si $s \neq s'$.

Soit B un ARP à la demande. Pour tous entiers strictement positifs t et w et pour toute configuration (s,q) , posons

$$f_t(s,q) = \begin{cases} 1 & \text{si } Y_t = (s,q) \text{ et si la transition de } Y_{t-1} \text{ à } Y_t \text{ est une} \\ & \text{transition avec défaut de page} \\ 0 & \text{sinon} \end{cases}$$

et :

$$N_w(s,q) = \sum_{t=1}^w f_t(s,q)$$

Définition 5

Le taux de défauts de page pour l'ARP B est

$$(4) \quad F(B) = \sum_{(s,q) \in S \times Q} \left(\lim_{w \rightarrow \infty} E \left\{ \frac{N_w(s,q)}{w} \right\} \right)$$

sous réserve que les limites existent.

L'algorithme B étant à la demande, le taux de transfert de page vers la mémoire rapide est égal à $F(B)$. En revanche le taux de transfert en sens inverse (taux de vidage) peut être inférieur. Nous allons l'évaluer en supposant qu'une page déchargée n'est effectivement recopiée en mémoire lente que

si elle a été modifiée. Désignant par (s', q') la configuration Y_{t-1} , nous posons

$$h_t(s, q) = \begin{cases} 1 & \text{si } Y_t = (s, q), f_t(s, q) = 1 \text{ et que la page qui appar-} \\ & \text{tient à } s' \text{ sans appartenir à } s \text{ est dans un emplacement} \\ & \text{i tel que } b_i^{t-1} = 1 \\ 0 & \text{sinon} \end{cases}$$

$$\text{et : } M_w(s, q) = \sum_{t=1}^w h_t(s, q)$$

Définition 6

Le taux de vidage de l'ARP B est

$$(5) \quad D(B) = \sum_{(s, q) \in S \times Q} \left(\lim_{w \rightarrow \infty} E \left\{ \frac{M_w(s, q)}{w} \right\} \right)$$

sous réserve que les limites existent.

Ayant restreint notre attention aux programmes dont le comportement est conforme au modèle à références indépendantes, nous allons établir une expression plus simple du taux de défauts de page. Nous appellerons configuration possible de l'ARP B toute configuration (s, q) satisfaisant l'une ou l'autre des conditions suivantes :

- . $(s, q) = (s_0, q_0)$
- . il existe une suite finie $(s_1, q_1), \dots, (s_{k-1}, q_{k-1}),$
 $(s_k, q_k) = (s, q)$ ayant la propriété que pour tout i

($1 \leq i \leq k$) il existe une page r telle que l'élément
 $(v(s_{i-1}, q_{i-1}), v(s_i, q_i))$ de la matrice $M(r)$ est
 strictement positif.

Nous désignerons par V l'ensemble des configurations possibles
 $(V \subset S \times Q)$ et nous supposons que la numérotation v est telle que

$$1 \leq v(s, q) \leq |V| \iff (s, q) \in V$$

(cette dernière hypothèse est évidemment compatible avec la restriction
 $v(s_0, q_0) = 1$). Nous désignerons par Z_t la variable aléatoire prenant ses
 valeurs dans V dont toute réalisation est égale à la réalisation de Y_t :

$$Z_t = Y_t.$$

Le processus stochastique $(Z_t)_{t \in \mathbb{N}}$ est évidemment une chaîne de Markov ;
 cette restriction de l'espace de valeurs a pour but d'éviter des cas triviaux
 de non-ergodicité de la chaîne $(Y_t)_{t \in \mathbb{N}}$

Si la chaîne de Markov $(Z_t)_{t \in \mathbb{N}}$ est ergodique, sa matrice de transition
 C admet un seul vecteur de probabilité ξ comme vecteur propre à gauche pour
 la valeur propre 1 :

$$(6) \quad \xi C = \xi$$

La i^{e} composante de ξ , $\xi(i)$, est la probabilité qu'en "régime stationnaire"
 l'ARP B soit dans la configuration (s, q) de numéro $v(s, q) = i$. Nous établis-
 sons dans l'annexe 1 le résultat ci-dessous.

Théorème 1

Soit B un ARP qui gère à la demande les pages d'un programme conforme au modèle à références indépendantes. Désignons par $T(i)$ l'ensemble des entiers j tels que la transition de la configuration de numéro i à la configuration de numéro j soit une transition avec défaut de page. Si la chaîne de Markov $(Z_t)_{t \in \mathbb{N}}$ est ergodique, le taux de défauts de page de l'ARP est

$$(6) \quad F(B) = \sum_{i=1}^{|V|} \xi(i) \sum_{j \in T(i)} c_{ij}$$

où $C = (c_{ij})$ est la matrice de transition de $(Z_t)_{t \in \mathbb{N}}$

Désignons par q_ℓ la probabilité que la page ℓ soit modifiée lorsqu'elle est référencée. Si ℓ appartient à N_c , q_ℓ est nul. En posant $q_\ell = 1$ pour toute page ℓ appartenant à N_d , on représente le système dans lequel les pages de données sont toujours recopiées en mémoire lente lors de leur remplacement : un tel transfert est nécessaire si un dispositif tel que le bit de modification n'est pas incorporé dans le hardware. La démonstration de la proposition ci-dessous est analogue à celle du théorème 1.

Proposition 1

Si q_ℓ est égal à 0 ou 1 selon que la page ℓ appartient à N_c ou N_d et si la chaîne de Markov $(Z_t)_{t \in \mathbb{N}}$ est ergodique, alors le taux de vidage de l'ARP à la demande B est égal à

$$(7) \quad D(B) = \sum_{i=1}^{|V|} \xi(i) \sum_{j \in T(i)} c_{ij} \gamma_{ij}$$

où nous avons posé

$$\gamma_{ij} = \begin{cases} 1 & \text{si la page faisant défaut lors de la transition de la} \\ & \text{configuration n° i à la configuration n° j est une pa-} \\ & \text{ge de } N_D \\ 0 & \text{sinon} \end{cases}$$

2.2 LES ALGORITHMES FIFO ET LRU

Proposition 2

Pour un programme ayant un comportement conforme au modèle à références indépendantes, le taux de défauts de page obtenu par l'algorithme FIFO est

$$(8) \quad F(\text{FIFO}) = \frac{\sum_{s \in S} \left(\sum_{i \in s} \beta_i \sum_{j \notin s} \beta_j \right)}{\sum_{s \in S} \sum_{i \in s} \beta_i}$$

(l'ensemble S des états de mémoire est défini dans l'exemple 2 du paragraphe 1.1)

Comme nous l'avons déjà remarqué, pour cet algorithme, l'ensemble V des configurations possibles est l'ensemble des couples (s,q) où q est un élément quelconque de Q et s la partie de N formée des pages qui sont des composantes de q.

Soit (s,q) la configuration possible de numéro i ; l'état de contrôle q est un m-uple que nous noterons $(p_1, p_2, \dots, p_m)$. Pour tout entier j compris entre 1 et $|V|$

$$c_{ij} = \sum_{r=1}^n m_{ij}^{(r)} \beta_r$$

Nous en déduisons

$$c_{ij} = \begin{cases} \beta_i & \text{si } i = j \\ \beta_r & \text{si la configuration de numéro } j \text{ a pour état de} \\ & \text{contrôle } (r, p_1, p_2, \dots, p_{m-1}) \\ 0 & \text{sinon} \end{cases}$$

Cette matrice est transitive (l'ARP peut passer d'une configuration possible à n'importe quelle autre en un nombre fini de transitions) et apériodique ($c_{ii} > 0$). La chaîne de Markov $(Z_t)_{t \in \mathbb{N}}$ est donc ergodique. On vérifie sans peine que la i^e composante du vecteur ξ est

$$\xi(i) = \xi(v(s, q)) = \frac{\prod_{p \in s} \beta_p}{\sum_{s \in S} \prod_{p \in s} \beta_p}$$

La proposition 2 apparaît donc comme un corollaire du théorème 1.

Ce résultat a déjà été obtenu par King (8), quoique de manière légèrement différente. King a également évalué le taux de défauts de page produit par l'algorithme LRU ; la formule qu'il a établie est peu utile car elle se compose d'une somme d'expressions indexées par Q dont le calcul effectif est presque impossible en pratique ($|Q| = \frac{n!}{(n-m)!}$). Nous établissons ci-dessous une autre formule où la sommation est étendue sur l'ensemble des parties de N : si $n = 15$ et $m = 8$, alors $\frac{n!}{(n-m)!} / 2^n \simeq 8000$.

Proposition 3

Considérons un programme dont le comportement est conforme au modèle à références indépendantes ; soit m la taille (en pages) de la zone qui lui est attribuée en mémoire rapide. Pour toute partie θ de N posons

$$(9) \quad x(\theta) = 1 - \sum_{p \in \theta} \beta_p$$

et

$$(10) \quad g_m(\theta) = \frac{x(\theta)^m}{1-x(\theta)} \sum_{\ell=0}^{n-m-1} (-1)^\ell \binom{|\theta|-1}{\ell} x(\theta)^{n-m-1-\ell}$$

Le taux de défauts de page provoqué par l'algorithme LRU est

$$(11) \quad F(\text{LRU}) = \sum_{\theta \in N^*} (g_m(\theta) \times \sum_{p \in \theta} \beta_p^2)$$

où N^* désigne l'ensemble des parties propres de N (c'est-à-dire des parties autres que N et $\emptyset$)

Démonstration

Supposons que la distribution de la chaîne de Markov des configurations possibles est stationnaire. Nous allons calculer la probabilité Ψ_j ($1 \leq j \leq n$) qu'à l'instant t (précisément : juste après la t^{e} référence) il y ait au moins m pages dont la dernière référence soit plus récente que la dernière référence de la page j . Autrement dit Ψ_j est la probabilité qu'à l'instant t le rang de la page j dans l'ordre LRU soit supérieur à m .

Il est clair que

$$(12) \quad F(\text{LRU}) = \sum_{j=1}^n \Psi_j \Pr[R_{t+1} = j] = \sum_{j=1}^n \Psi_j \beta_j$$

Soient i et T deux entiers tels que $1 \leq i \leq n-1$ et $T \geq 1$.

Nous allons calculer la probabilité $\psi(j, i, T)$ que les évènements ci-dessous se réalisent tous trois :

- . la page j est référencée à l'instant $t - T$
- . la page j n'est pas référencée dans l'intervalle
 $\{ t-T+1, t-T+2, \dots, t \}$
- . à l'instant t , la page j est au rang $(i+1)$ dans l'ordre LRU

De cette définition nous tirons ;

$$(13) \quad \psi(j, i, T) = \Pr [R_{t-T} = j] \times \Pr [(\forall \ell)(0 \leq \ell \leq T-1) : R_{t-\ell} \neq j] \times \\ \Pr [i \text{ pages exactement sont référencées de } t-T+1 \text{ à } t | \\ (\forall \ell)(0 \leq \ell \leq T-1) : R_{t-\ell} \neq j]$$

Les deux premières probabilités figurant dans ce produit sont respectivement égales à β_j et $(1 - \beta_j)^T$. La troisième est la probabilité que la taille d'un ensemble de travail $W(t, T)$ (voir définition 1, chapitre 4) soit i pour un programme de $(n-1)$ pages dont les probabilités d'être référencées sont respectivement $\frac{\beta_1}{1-\beta_j}, \frac{\beta_2}{1-\beta_j}, \dots, \frac{\beta_{j-1}}{1-\beta_j}, \frac{\beta_{j+1}}{1-\beta_j}, \dots, \frac{\beta_n}{1-\beta_j}$.

Introduisons trois nouvelles notations :

- $N(j, i)$ est l'ensemble des parties de N qui contiennent $(i+1)$ éléments dont la page j
- $A_r(T, j) = \sum_{*} \left(1 - \frac{\beta_{p1}}{1-\beta_j} - \frac{\beta_{p2}}{1-\beta_j} - \dots - \frac{\beta_{p_r}}{1-\beta_j} \right)^T$

où $\sum_{*}$ désigne une sommation effectuée sur toutes les parties $\{p_1, p_2, \dots, p_r\}$ à r éléments de $N - \{j\}$. (en particulier :

$$A_0(T, j) = 1)$$

$$-\zeta_i(T) \text{ vaut } 1 \text{ si } T \geq i, \quad 0 \text{ sinon.}$$

D'après le théorème trois du chapitre 4, le troisième facteur du membre de droite de l'égalité (13) est égal à

$$(14) \quad \sum_{r=n-i-1}^{n-2} (-1)^{r-n-i+1} \binom{r}{n-i-1} A_r(T, j) \zeta_i(T)$$

Pour tout entier r ($1 \leq r \leq n-2$) posons

$$(15) \quad \begin{aligned} B_r(T, j) &= A_r(T, j) (1 - \beta_j)^T \\ &= \sum_{\theta \in N(j, r)} x(\theta)^T \end{aligned}$$

ce qui nous permet d'écrire :

$$\psi(j, i, T) = \beta_j \sum_{r=n-i-1}^{n-2} (-1)^{r-n-i-1} \binom{r}{n-i-1} B_r(T, j) \zeta_i(T)$$

La probabilité que la dernière référence à la page j avant t ait lieu à $t - T$ et que le rang de j dans l'ordre LRU soit strictement supérieur à m est égale à $\sum_{i=m}^{n-1} \psi(j, i, T)$

c'est-à-dire à :

$$\begin{aligned}
 &= \beta_j \sum_{\theta \in N(j,k)} \sum_{\ell=0}^{n-m-1} (-1)^{k+\ell} \binom{k}{\ell} \frac{x(\theta)^{n-1-\ell}}{1-x(\theta)} \\
 &= \beta_j \sum_{\theta \in N(j,k)} \frac{x(\theta)^m}{1-x(\theta)} (-1)^k \sum_{\ell=0}^{n-m-1} (-1)^\ell \binom{k}{\ell} x(\theta)^{n-m-1-\ell}
 \end{aligned}$$

Compte tenu de l'égalité (10), nous pouvons donc écrire :

$$\psi_j = \beta_j \sum_{k=0}^{n-2} \sum_{\theta \in N(j,k)} g_m(\theta)$$

et, d'après (12)

$$F(LRU) = \sum_{j=1}^n \sum_{k=0}^{n-2} \sum_{\theta \in N(j,k)} g_m(\theta) \beta_j^2$$

En réordonnant cette sommation pour grouper tous les termes relatifs à une même partie θ , nous obtenons

$$F(LRU) = \sum_{\theta \in N^*} (g_m(\theta) \sum_{p \in \theta} \beta_p^2)$$

ce qui est le résultat annoncé.

3 - LES ALGORITHMES DE REMPLACEMENT DE PAGE ALEATOIRES A PRECHARGEMENT PARTIEL

(ARPAPP).

De manière informelle un ARPAPP est un ARP à la demande ayant les deux propriétés suivantes :

- Une partie ϕ_k à k éléments ($k < m$) de l'espace d'adresse N du programme est toujours résidente en mémoire (préchargement partiel)

- lors d'un défaut de page chacune des $\ell (=m-k)$ pages chargées en mémoire mais n'appartenant pas à ϕ_k a la probabilité $1/\ell$ d'être désignée comme page à remplacer (ARP aléatoire).

Nous allons définir précisément les ARPAPP en utilisant les notions introduites dans les paragraphes précédents. Nous donnerons ensuite un aperçu des propriétés de cette classe d'algorithmes, les programmes manipulés étant supposés avoir un comportement à références indépendantes.

Cette classe d'algorithmes, introduite par Gelenbe qui en établit les principales propriétés [7], est spécialement intéressante pour l'évaluation des ARP. D'une part elle contient l'algorithme optimal A_0 [10] et l'algorithme RAND. D'autre part un sous-ensemble d'ARPAPP, la classe OPT, constitue une hiérarchie d'algorithmes qui peuvent servir de points de repère pour la comparaison des ARP utilisés en pratique. Au chapitre trois, nous utiliserons les résultats sur le comportement des ARPAPP pour analyser une nouvelle méthode de gestion de mémoire centrale et de tambour que nous proposons.

3.1 DEFINITION ET TAUX DE DEFAULTS DE PAGE

Définition 7

Soient k un entier tel que $0 \leq k \leq m-1$ et ϕ_k une partie à k éléments de N . L'ARP aléatoire à préchargement partiel de ϕ_k (ARPAPP) est un ARP ayant les trois propriétés suivantes :

a) $S = \{s \mid s \in S_m \text{ et } \phi_k \subset s\}$

b) $Q = \{q_0\}$

c) Pour tous s et s' appartenant à S et toute page r de N , les entiers i et j étant respectivement égaux à $v(s, q_0)$ et $v(s', q_0)$:

$$m_{ij}(r) = \begin{cases} 1 & \text{si } i = j \text{ et } r \in s \\ \frac{1}{m-k} & \text{si } r \in s', r \notin s \text{ et que toute autre page de } s' \text{ appartient à } s \\ 0 & \text{sinon} \end{cases}$$

Un ARPAPP a donc un seul état de contrôle et par suite les matrices $M(r)$ sont d'ordre $|S|$. Nous appellerons k l'ordre de l'ARPAPP. La classe des ARPAPP d'ordre k a $\binom{n}{k}$ éléments. Remarquons que ces algorithmes sont des ARP à la demande.

Théorème 2

Considérons un programme dont l'ensemble des pages est $N = \{1, 2, \dots, n\}$ et dont le comportement est représenté par le modèle à références indépendantes, la probabilité de référencer la page i étant notée β_i ($i = 1, 2, \dots, n$). Soit B un ARPAPP maintenant en mémoire les pages de la partie $\phi_k = \{1, 2, \dots, k\}$ de N . Le taux de défauts de page de cet algorithme est

$$(18) \quad F(B) = \frac{\sum_{s \in S} \left[\prod_{p \in s - \phi_k} \beta_p \times \sum_{j \notin s} \beta_j \right]}{\sum_{s \in S} \left[\prod_{p \in s - \phi_k} \beta_p \right]}$$

Ce théorème est démontré au paragraphe 3.2. Il en est de même du théorème 3, ci-dessous, relatif au taux de vidage.

Théorème 3

Considérons un programme conforme au modèle à références indépendantes

et dont l'espace d'adresse est partagé en un ensemble de pages de données, N_d , et un ensemble de pages de code, N_c . Lorsqu'elles sont remplacées, les pages de N_d sont toujours recopiées en mémoire lente ; celles de N_c ne le sont jamais. Pour tout état s de la mémoire, posons

$$(19) \quad K(s) = |(s - \phi_k) \cap N_d|$$

Le taux de vidage de l'ARPAPP B est

$$(20) \quad D(B) = \frac{\sum_{s \in S} \left[\prod_{p \in s - \phi_k} \beta_p \times \sum_{j \notin s} \beta_j \right]}{\sum_{s \in S} \left[\prod_{p \in s - \phi_k} \beta_p \right]}$$

3.2. DEMONSTRATION DES THEOREMES 2 ET 3

Soient (s, q_0) et (s', q_0) deux configurations de l'ARP B. La probabilité que la chaîne de Markov $(Y_t)_{t \in \mathbb{N}}$ passe de l'une à l'autre en un nombre fini de transitions est strictement positive. Supposons en effet que s' contienne exactement d pages qui n'appartiennent pas à s : $j_1, j_2, \dots, j_d$; la probabilité que la chaîne de Markov $(Y_t)_{t \in \mathbb{N}}$ passe de l'état (s, q_0) à l'état (s', q_0) en d transitions est au moins égale à $\frac{\beta_{j_1} \beta_{j_2} \dots \beta_{j_d}}{(m - k)^d}$. Ceci implique que

- tous les états sont possibles et, par conséquent, les chaînes de Markov (Y_t) et (Z_t) sont identiques.
- la chaîne de Markov (Y_t) est transitive.

En outre la chaîne de Markov (Y_t) est apériodique puisque

$$\Pr [Y_t = (s, q_0) \mid Y_{t-1} = (s, q_0)] = \Pr [R_t \in s] > 0$$

Elle est donc ergodique, ce qui va nous permettre d'utiliser le théorème 1 et la proposition 1.

Calcul des coefficients de la matrice C

Soient s et s' deux états de la mémoire, $i = v(s, q_0)$ et $j = v(s', q_0)$ les numéros des configurations correspondantes. Il résulte de la définition 7 :

$$c_{ij} = \begin{cases} \sum_{r=1}^n \beta_r & m_{ii}(r) = \sum_{r \in s} \beta_r & \text{si } i = j \\ \frac{\beta_r}{m-k} & \text{s'il existe une page } r & \text{(unique) de } s' \text{ qui n'appartient pas à } s \\ 0 & \text{sinon} \end{cases}$$

Calcul de la distribution stationnaire de $(Y_t)_{t \in \mathbb{N}}$

Soit ξ le vecteur de probabilité tel que

$$(21) \quad \xi \cdot C = \xi$$

Cette équation équivaut au système d'équations :

$$(22) \quad \xi(i) = \sum_{j=1}^{|S|} \xi(j) c_{ji} \quad (i = 1, 2, \dots, |S|)$$

Nous allons montrer que la solution en est, pour tout entier $i=v(s,q_0)$:

$$(23) \quad \xi(i) = \frac{\beta_{p_{k+1}} \beta_{p_{k+2}} \dots \beta_{p_m}}{\sum_{s \in S} \beta_{p_{k+1}} \beta_{p_{k+2}} \dots \beta_{p_m}} \quad (1 \leq i \leq |S|)$$

l'état de mémoire s étant noté $\{ 1, 2, \dots, k, p_{k+1}, \dots, p_m \}$ si $k \geq 1$

$\{ p_1, p_2, \dots, p_m \}$ si $k = 0$

A l'entier $i = v(s, q_0)$, associons l'ensemble, noté $R(i, f)$ ($1 \leq f \leq m-k$) des numéros j ($j \neq i$) des configurations d'où la chaîne de Markov (Y_t) peut accéder à i en une seule transition causée par une référence à la page p_{k+f} (cette transition est une transition avec défaut de page puisque $i \neq j$). Autrement dit :

$$(24) \quad R(i, f) = \{ j = v(s', q_0) \mid (\exists a \in N-s) : s' = s - \{p_{k+f}\} \cup \{a\} \}$$

L'équation (23) peut alors s'écrire :

$$(25) \quad \xi(i) = \xi(i) \sum_{r \in s} \beta_r + \sum_{f=1}^{m-k} \sum_{j \in R(i, f)} \xi(j) \frac{\beta_{p_{k+f}}}{m-k}$$

Pour montrer que (23) en décrit la solution, remarquons que si j appartient à $R(i, f)$, $\xi(j) = \frac{\beta_a}{\beta_{p_{k+f}}} \xi(i)$. Il suffit donc de vérifier que

$$(26) \quad \xi(i) = \xi(i) \sum_{r \in s} \beta_r + \sum_{f=1}^{m-k} \sum_{a \notin s} \beta_a \frac{\xi(i)}{m-k}$$

c'est-à-dire que

$$\xi(i) = \xi(i) \sum_{r \in s} \beta_r + \xi(i) \sum_{a \notin s} \beta_a$$

Cette dernière égalité est évidemment correcte.

Calcul du taux de défauts de page

Conformément au théorème 1, il nous reste à évaluer $\sum_{j \in T(i)} c_{ij}$. De sa définition il résulte que $T(i)$ est l'ensemble des $j = v(s', q_0)$ tels que

$$(\exists r' \in N - s) (\exists r \in s - \phi_k) : s' = s \cup \{r'\} - \{r\}$$

La chaîne de Markov $(Y_t)_{t \in \mathbb{N}}$ passe de l'état i à l'état j avec probabilité $\frac{\beta_{r'}}{m-k}$. Nous pouvons donc écrire :

$$(27) \quad \sum_{j \in T(i)} c_{ij} = \sum_{r' \in N-s} \sum_{r \in s-\phi_k} \frac{\beta_{r'}}{m-k} = \sum_{r' \in N-s} \beta_{r'}$$

Le théorème 2 est donc une conséquence du théorème 1 et des égalités (23) et (27).

Calcul du taux de vidage

En gardant les notations de l'alinéa ci-dessus, nous obtenons

$$\begin{aligned} \sum_{j \in T(i)} c_{ij} \gamma_{ij} &= \sum_{r' \in N-s} \sum_{r \in (s-\phi_k) \cap N_d} \frac{\beta_{r'}}{m-k} \\ &= \sum_{r' \in N-s} K(s) \frac{\beta_{r'}}{m-k} \end{aligned}$$

ce qui, compte tenu de la proposition 1 et de l'égalité (23) achève la démonstration du théorème 3.

3.3 EQUIVALENCE DE FIFO ET RAND

L'algorithme de remplacement de page RAND est l'ARPAPP d'ordre $k = 0$. On déduit du théorème 2 le corollaire ci-dessous.

Proposition 4

Pour tout programme ayant un comportement conforme au modèle à références indépendantes, le taux de défauts de page obtenu par l'algorithme RAND est

$$(28) \quad F(\text{RAND}) = \frac{\sum_{s \in S} \left(\prod_{p \in s} \beta_p \times \sum_{j \notin s} \beta_j \right)}{\sum_{s \in S} \left(\prod_{p \in s} \beta_p \right)}$$

La comparaison des propositions 2 et 4 conduit au théorème d'équivalence suivant (Gelenbe [7])

Théorème 4

Pour tout programme ayant un comportement à références indépendantes,

$$(29) \quad F(\text{FIFO}) = F(\text{RAND})$$

Ce résultat appelle un commentaire. Lorsque le choix des pages à remplacer est effectué par RAND, le taux de défauts de page est une fonction décroissante de la taille m de la zone de mémoire rapide attribuée au

programme (supposé conforme au modèle à références indépendantes) : ce fait est établi dans [11] en particulier. Or il a été montré [12] que pour certaines chaînes de références l'augmentation de m peut conduire à une augmentation du taux de défauts de page lorsque l'ARP utilisé est FIFO. Il résulte du théorème 4 qu'il est relativement rare qu'une de ces chaînes paradoxales soit engendrée par le modèle de comportement de programmes que nous avons utilisé.

3.4. LES ALGORITHMES OPTIMAUX

Parmi les ARPAPP d'ordre k , il en est un - que nous noterons $B_0(k,m)$ - pour lequel le taux de défauts de page est minimum : c'est celui qui maintient en mémoire l'ensemble ϕ_k formé des pages de N ayant les plus fortes probabilités d'être référencées. Ce résultat (qui ne choque pas l'intuition) est démontré dans [11]. Nous désignerons par OPT cette sous-classe d'ARPAPP. Son utilisation pour la comparaison d'ARP est rendue possible par les deux propriétés suivantes :

$$(\forall k) (0 \leq k \leq m-2) : F(B_0(k+1,m)) \leq F(B_0(k,m))$$

$$(\forall m) (k+2 \leq m \leq n-1) : F(B_0(k,m)) \leq F(B_0(k, m-1))$$

L'algorithme $B_0(m-1,m)$ est souvent noté A_0 . Aho, Denning et Ullman [10] ont montré qu'il est optimal dans la classe des ARP tels qu'ils sont définis au paragraphe 1.1, pourvu que le modèle de comportement de programme soit à références indépendantes. Son taux de défauts de page est donné ci-dessous.

Proposition 5

Supposons la numérotation des pages telle que

$$\beta_1 \geq \beta_2 \geq \dots \geq \beta_{m-1} \geq \beta_m \geq \dots \geq \beta_n$$

Le taux de défauts de page moyen de A_0 est

$$F(A_0) = \frac{\sum_{i=m}^n \beta_i (1 - \beta_1 - \dots - \beta_{m-1} - \beta_i)}{\sum_{i=m}^n \beta_i}$$

Il convient de faire attention aux hypothèses sous lesquelles on peut affirmer que A_0 est optimal. En particulier cette propriété peut disparaître si l'on considère d'autres ARP que ceux de la définition 1. Belady a proposé l'algorithme MIN qui lors d'un défaut de page remplace la page pour laquelle la prochaine référence est la plus éloignée [13] : l'algorithme MIN n'est pas un ARP au sens où nous l'entendons puisqu'il ne se contente pas d'une quantité finie d'informations sur le comportement passé du programme. A titre de contre-exemple nous calculons dans l'annexe B la probabilité $F(\text{MIN})$ d'une référence avec défaut de page pour un programme de 3 pages dont 2 sont présentes en mémoire rapide, le remplacement étant décidé par MIN.

Nous obtenons

$$(31) \quad F(\text{MIN}) = \frac{2}{\frac{1}{\beta_1} + \frac{1}{\beta_2} + \frac{1}{\beta_3}} < \frac{2}{\frac{1}{\beta_1} + \frac{1}{\beta_2}} = F(A_0)$$

(on a supposé : $\beta_1 \geq \beta_2 \geq \beta_3$)

4 - EVALUATION DES ARP SUR LE MODELE DE COMPORTEMENT DE PROGRAMME A LISTE LRU

Soit $\mathcal{P}$ l'ensemble des permutations de l'ensemble d'adresse N . Rappelons que si le programme est conforme au modèle à liste LRU, son état à l'instant t est un élément σ_t de $\mathcal{P}$. La suite

$$\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_t, \dots$$

est la réalisation d'une chaîne de Markov dont nous désignerons par

$R = (r_{\sigma, \sigma'})$ la matrice de transition,

Soit $B = (N, S, Q, M(r), s_0, q_0)$ un ARP qui gère les pages de ce programme. Nous désignerons par $Y_t = (s_t, q_t)$ la configuration de B à l'instant t . Le processus stochastique $(\tilde{Y}_t)_{t \in \mathbb{N}}$ défini par

$$(32) \quad \tilde{Y}_t = (\sigma_t, s_t, q_t)$$

est une chaîne de Markov à valeurs dans $\mathcal{P} \times S \times Q$. Soient $i = (\sigma, s, q)$ et $j = (\sigma', s', q')$ deux éléments de $\mathcal{P} \times S \times Q$; nous poserons $k = v(s, q)$, $k' = v(s', q')$ et $q' = (a_1, a_2, \dots, a_n)$.

La probabilité que la chaîne de Markov $(\tilde{Y}_t)$ passe en une seule transition de l'état i à l'état j est

$$\tilde{c}_{ij} = \begin{cases} r_{\sigma\sigma'} m_{kk'}(a_1) & \text{si } a_1 \in s' \\ 0 & \text{sinon.} \end{cases}$$

Comme au paragraphe 2.1. nous définissons l'ensemble $\tilde{V}$ des états possibles c'est-à-dire la partie de $\mathcal{J} \times S \times Q$ formée des valeurs que la chaîne $(\tilde{Y}_t)$ peut atteindre à partir de l'état initial (σ_0, s_0, q_0) . Remarquons qu'une condition nécessaire pour que l'état (σ, s, q) soit possible est que la première composante du n-uple σ soit un élément de s (la dernière page référencée doit évidemment être présente en mémoire rapide). Nous désignons par $(\tilde{Z}_t)_{t \in \mathbb{N}}$ la co-restriction de la chaîne $(\tilde{Y}_t)_{t \in \mathbb{N}}$ à l'ensemble $\tilde{V}$. Le résultat ci-dessous est l'analogue du théorème 1 et se démontre de la même manière.

Théorème 5

Soit B un ARP qui gère à la demande les pages d'un programme conforme au modèle à liste LRU. Soit $i = (\sigma, s, q)$ un élément de $\tilde{V}$: nous désignerons par $\tilde{T}(i)$ l'ensemble des éléments $j = (\sigma', s', q')$ de $\tilde{V}$ tels que l'ARP B passe de la configuration (s, q) à la configuration (s', q') en une transition avec défaut de page. Si la chaîne de Markov $(\tilde{Z}_t)$ est ergodique, désignons par $\xi(i)$ la probabilité stationnaire de l'état possible i ;
on a alors

$$(33) \quad F(B) = \sum_{i \in \tilde{V}} \xi(i) \sum_{j \in \tilde{T}(i)} \tilde{c}_{ij}$$

A titre d'illustration de ce théorème nous montrerons le résultat ci-dessous.

Proposition 6

Si l'ARP LRU gère les pages d'un programme conforme au modèle à liste LRU (la page à la distance i étant référencée avec probabilité α_i), le taux de défauts de page est

$$(34) \quad P(LRU) = \sum_{\ell=m+1}^n \alpha_{\ell}$$

Démonstration

Un état (σ, s, q) est possible si et seulement si les conditions ci-dessous sont toutes deux réalisées

- . le m-uple q s'obtient par troncation de σ
- . l'ensemble s est formé des composantes de q

Nous pouvons donc écrire : $\tilde{V} \approx \mathcal{P}$ et $(\tilde{Z}_t) \approx (\sigma_t)$

De l'étude de la chaîne de Markov (σ_t) qui a été faite au chapitre premier, il résulte que $(\tilde{Z}_t)$ est ergodique et que pour tout état i , $\tilde{\xi}(i) = \frac{1}{n!}$.

Comme

$$\sum_{j \in \tilde{T}(i)} \tilde{c}_{ij} = \sum_{\ell=m+1}^n \alpha_{\ell} \quad \text{et que} \quad |\tilde{V}| = n!, \quad \text{la formule (34) est}$$

une conséquence directe du théorème 5.

ANNEXE A

DEMONSTRATION DU THEOREME 1

A la chaîne de Markov $(Z_t)_{t \in \mathbb{N}}$, associons les variables aléatoires

$h_t(.)$ et $m_w(.)$ ($t, w \in \mathbb{N}$) définies pour tout $i \in \mathbb{N}$ par

$$h_t(i) = \begin{cases} 1 & \text{si } v(Z_t) = i \\ 0 & \text{sinon} \end{cases}$$

et

$$m_w(i) = \sum_{t=0}^{w-1} h_t(i)$$

Si la chaîne de Markov $(Z_t)_{t \in \mathbb{N}}$ est ergodique, d'après la loi des grands nombres pour les chaînes de Markov (théorème 4.2.1 in [6]), nous obtenons :

$$(A1) \quad \lim_{w \rightarrow \infty} E \left(\frac{m_w(i)}{w} \right) = \xi(i)$$

Pour tout entier t strictement positif, posons

$$T_t(i, j) = \begin{cases} 1 & \text{si } v(Z_{t-1}) = i \text{ et } v(Z_t) = j \\ 0 & \text{sinon} \end{cases}$$

et

$$(A2) \quad \begin{aligned} e_w(i, j) &= E \left(\frac{\sum_{t=1}^w T_t(i, j)}{w} \right) \\ &= \frac{1}{w} \sum_{t=1}^w E(T_t(i, j)) \end{aligned}$$

Nous remarquons que $E(T_t(i, j)) = \Pr [T_t(i, j) = 1]$

et puisque

$$\Pr [T_t(i, j) = 1] = \Pr [v(Z_t) = j \mid v(Z_{t-1}) = i] \cdot \Pr [h_{t-1}(i) = 1]$$

nous en déduisons :

$$e_w(i, j) = \frac{1}{w} c_{ij} \sum_{t=0}^{w-1} \Pr [h_t(i) = 1] = c_{ij} E\left(\frac{m_w(i)}{w}\right)$$

Il résulte de l'application de la loi des grands nombres (égalité (A1)) :

$$(A3) \quad \lim_{w \rightarrow \infty} e_w(i, j) = \xi(i) c_{ij}$$

Pour calculer $F(B)$, nous avons besoin d'évaluer la moyenne

$$N_w(s, q) = \sum_{t=1}^w \sum_{\substack{i \text{ tel} \\ \text{que } j \in T(i)}} T_t(i, j)$$

où $j = v(s, q)$.

D'après l'égalité (A2) :

$$E\left(\frac{N_w(s, q)}{w}\right) = \sum_{\substack{i \text{ tel} \\ \text{que} \\ j \in T(i)}} e_w(i, j)$$

Par conséquent, le taux de défauts de page $F(B)$ qui est défini comme étant

$$F(B) = \sum_{(s, q) \in V} \lim_{w \rightarrow \infty} E\left(\frac{N_w(s, q)}{w}\right)$$

vaut :

$$\sum_{j=1}^{|V|} \sum_{\substack{i \text{ tel que} \\ j \in T(i)}} \xi(i) c_{ij} = \sum_{i=1}^{|V|} \sum_{j \in T(i)} \xi(i) c_{ij}$$

Ceci achève la démonstration du théorème.

ANNEXE B

CALCUL DE F (MIN)

Nous allons tout d'abord donner une définition du taux de défauts de page qui puisse s'appliquer à MIN. Supposons que durant l'exécution d'une chaîne de références représentée par le processus stochastique $(R_t)_{t \in \mathbb{N}}$ des défauts de page se produisent aux instants $t_0, t_1, \dots, t_n, \dots$ ($t_{n+1} > t_n$) qui sont des réalisations d'un processus $(T_n)_{n \in \mathbb{N}}$.

Posons :

$$(A1) \quad E_0 = T_0$$

$$(A2) \quad E_n = T_n - T_{n-1}$$

Nous définissons le taux de défauts de page F comme l'inverse de la durée de vie moyenne. Plus précisément :

$$(A3) \quad \frac{1}{F} = \lim_{w \rightarrow \infty} E \left(\frac{E_0 + E_1 + \dots + E_{w-1}}{w} \right)$$

Appliquée aux ARP tels qu'ils sont définis au paragraphe 1.1. cette définition coïncide avec la définition 5. Nous l'admettrons.

Nous allons calculer F(MIN) pour un programme de trois pages, conforme au modèle à références indépendantes, et auquel sont attribuées deux pages physiques en mémoire rapide. Par convention d'écriture, tout au long de cette annexe les entiers i, j, k seront supposés compris entre 1 et 3 et deux à deux distincts. Considérons le processus stochastique $(D_n)_{n \in \mathbb{N}}$ ainsi défini :

$d_n = (i, j) \iff$ juste avant l'instant t_n du n^e défaut de page, la page i n'est pas en mémoire et, des deux pages j et k , j est celle qui est référencée le plus tôt après t_n .

Si $d_n = (i, j)$, le n^e défaut de page est donc provoqué par une référence à i : $r_{t_n} = i$. Il est essentiel de remarquer que, pour évaluer la seconde composante de d_n , il suffit d'examiner les références

$$r_{t_n+1}, r_{t_n+2}, \dots, r_{t_{n+1}-1}$$

La valeur prise par D_n (et a fortiori celles de $D_{n-1}, D_{n-2}, \dots$) ne reflète donc aucune information sur la chaîne de références au delà du $(n+1)^e$ défaut de page. Il en résulte que le processus $(D_n)_{n \in \mathbb{N}}$ est une chaîne de Markov à six états. Nous allons calculer ses probabilités de transition. Pour cela considérons la variable aléatoire $X_n(i)$ ($i = 1, 2, 3$) définie par :

la première référence à la page i après t_n a lieu à l'instant $t_n + x_n(i)$; si $r_{t_n} = i$, $x_n(i) = 0$

Il est clair que

$$\begin{aligned} \Pr[X_n(j) < X_n(k) \mid X_n(i) = 0] &= \sum_{x=1}^{\infty} \Pr[(\forall u) (1 \leq u \leq x-1) : R_{t+u} = i] \times \\ &\quad \Pr[R_{t+x} = j] \\ &= \sum_{x=1}^{\infty} \beta_i^{x-1} \beta_j = \frac{\beta_j}{1 - \beta_i} \end{aligned}$$

Cette formule préliminaire étant établie, remarquons que les deux composantes de l'état d_n sont les pages présentes en mémoire juste après t_{n-1} donc juste avant t_n .

La chaîne de Markov $(D_n)_{n \in \mathbb{N}}$ ne peut donc entrer dans l'état $(i,*)$ qu'à partir de (j,k) ou (k,j) . Les seules probabilités de transition non nulles sont donc

$$\begin{aligned} \Pr [d_n = (i, j) | d_{n-1} = (k, j)] &= \Pr [d_n = (i, j) | d_{n-1} = (j, k)] \\ &= \Pr [X_n(j) < X_n(k) | X_n(i) = 0] = \frac{\beta_j}{1-\beta_i} \end{aligned}$$

(pour toutes les permutations (i, j, k) de $(1, 2, 3)$)

En outre la chaîne de Markov $(D_n)_{n \in \mathbb{N}}$ est

- transitive puisque son graphe de transitions possède un circuit hamiltonien :

$$(1,2) (3,1) (2,1) (3,2) (1,3) (2,3) (1,2)$$

- apériodique puisqu'il existe deux circuits de longueurs premières entre elles :

$$(1,2) (3,2) (1,2) \quad \text{de longueur 2}$$

$$(1,2) (3,1) (2,3) (1,2) \quad \text{de longueur 3}$$

Elle est donc ergodique. La distribution stationnaire $\{\pi(i,j)\}$ est solution du système d'équations :

$$(A4) \quad \pi(i,j) = (\pi(j,k) + \pi(k,j)) \cdot \frac{\beta_j}{1-\beta_i}$$

(pour toute permutation de $(1, 2, 3)$)

On en déduit sans peine que pour tout état (i, j)

$$(A5) \quad \pi(i, j) = \frac{\beta_j}{2}$$

Nous allons maintenant calculer l'espérance de E_{n+1} conditionnée par la valeur de D_n :

$$(A6) \quad E(E_{n+1} \mid d_n = (i, j)) = E(X_n(k) \mid X_n(j) < X_n(k) ; X_n(i) = 0)$$

Evaluons la distribution de $X_n(k)$ ainsi conditionnée en considérant tout d'abord la distribution conjointe :

$$\Pr [X_n(k) = x ; X_n(j) < X_n(k) ; X_n(i) = 0] =$$

$$\sum_{y=1}^{x-1} \Pr[(\forall u) (1 \leq u \leq y-1) : r_{t_n+u} = i] \times \Pr[r_{t_n+y} = j] \times$$

$$\Pr[(\forall u) (y+1 \leq u \leq x-1) : r_{t_n+u} \in \{i, j\}] \times \Pr[r_{t_n+x} = k]$$

$$= \sum_{y=1}^{x-1} \beta_i^{y-1} \times \beta_j \times (\beta_i + \beta_j)^{x-y-1} \times \beta_k$$

$$= (\beta_i + \beta_j)^{x-2} \beta_j \beta_k \sum_{y=1}^{x-1} \left(\frac{\beta_i}{\beta_i + \beta_j} \right)^{y-1}$$

$$= \beta_k [(\beta_i + \beta_j)^{x-1} - \beta_i^{x-1}]$$

Donc :

$$\Pr [X_n(k) = x \mid X_n(j) < X_n(k) ; X_n(i) = 0] =$$

$$\frac{\beta_k [(\beta_i + \beta_j)^{x-1} - \beta_i^{x-1}]}{\sum_{x=1}^{\infty} \beta_k [(\beta_i + \beta_j)^{x-1} - \beta_i^{x-1}]} =$$

$$\frac{\beta_k (1-\beta_i)}{\beta_j} [(\beta_i + \beta_j)^{x-1} - \beta_i^{x-1}]$$

Il résulte donc de l'égalité (A6) :

$$E (E_{n+1} \mid d_n = (i, j)) = \sum_{x=1}^{\infty} \frac{\beta_k (1-\beta_i)}{\beta_j} x [(\beta_i + \beta_j)^{x-1} - \beta_i^{x-1}]$$

$$= \frac{\beta_k (1-\beta_i)}{\beta_j} \left[\frac{1}{\beta_k^2} - \frac{1}{(1-\beta_i)^2} \right] = \frac{1}{\beta_j} \left[\frac{\beta_j + \beta_k}{\beta_k} - \frac{\beta_k}{\beta_j + \beta_k} \right]$$

$$= \frac{1}{\beta_k} + \frac{1}{\beta_j + \beta_k}$$

Il résulte de la définition du taux de défauts de page (égalité (A3))

et de la loi des grands nombres pour les chaînes de Markov que

$$\frac{1}{F(\text{MIN})} = \sum_{(i,j,k)} E(E_{n+1} \mid d_n = (i,j)) \pi(i,j)$$

$$(A7) \quad = \sum_{(i,j,k)} \frac{1}{2} \left(\frac{\beta_j}{\beta_k} + \frac{\beta_j}{\beta_j + \beta_k} \right)$$

$$\text{Or} \quad \sum_{(i,j,k)} \frac{\beta_j}{\beta_j + \beta_k} = \sum \frac{\beta_j + \beta_k}{\beta_j + \beta_k} = 3$$

$$\text{et} \quad \sum_{(i,j,k)} \frac{\beta_j}{\beta_k} = \sum \frac{\beta_i + \beta_j}{\beta_k} = \sum_{k=1}^3 \frac{1-\beta_k}{\beta_k} = \sum_{k=1}^3 \frac{1}{\beta_k} - 3$$

Nous obtenons donc finalement

$$F(\text{MIN}) = \frac{2}{\frac{1}{\beta_1} + \frac{1}{\beta_2} + \frac{1}{\beta_3}}$$

ce qui est le résultat annoncé.

REFERENCES

- [1] MEINADIER, J.P. : Structure et fonctionnement des ordinateurs -
Larousse, Paris (1971)
- [2] SPANIOL, O. : Lokali tätsmodelle und Demand - Prepaging - Strategien -
Rapport de recherche A74 - 04, Universität des Saarlandes,
Saarbrücken.
- [3] KRAKOWIAK, S. : Conception et réalisation de systèmes à accès
multiple : allocation de ressources - Thèse de doctorat-ès-
sciences, Paris (1973)
- [4] ALEXANDER, M.T. : Time-sharing supervisor programs - University of
Michigan Engineering Summer Conference on Advanced Topics in
Systems Programming, Ann Arbor (juin-juillet 1971)
- [5] LIPTAY, J.S. : Structural aspect of the system 360 model 85 : the
cache - IBM Systems Journal, vol 7, 15 - 21 (1968)
- [6] KEMENY, J.C., SNELL, J.L. : Finite Markov chains - Van Nostrand,
Princeton, N.J. (1960)
- [7] GELENBE, E. : An unified approach to the evaluation of a class
of replacement algorithms - IEEE Trans. on Computers,
vol C-22, 6, 611-618 (1973)
- [8] KING III, W.F. : Analysis of demand paging algorithms -
IFIP Congress 1971, North Holland P.C.
- [9] GELENBE, E. : Modèles de Comportement de systèmes informatiques -
Thèse de doctorat-ès-sciences, Paris (1973)

- [10] AHO, A.V., DENNING, P.J., ULLMAN, J.D. : Principles of demand
paging algorithms - Journal . ACM18, 1, 80-93 (1971)

- [11] VANTILBORGH, H. : On random partially preloaded page replacement
algorithms - Rapport de recherche R 202, MBLÉ,
Bruxelles (1972)

- [12] BELADY, L.A., NELSON, R.A., SHELDER, G.S. : An anomaly in
space-time characteristics of certain programs -
Comm. ACM 12, 6, 349-353 (1969)

- [13] BELADY, L.A. : A study of replacement algorithms for a virtual
storage computer - IBM Sytem. J. 5, 2 (1966)

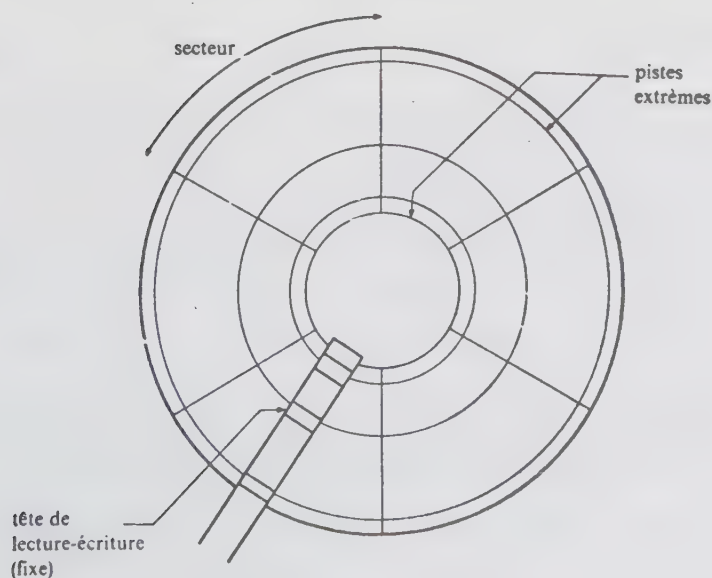
CHAPITRE TROIS

GESTION GLOBALE DE LA MEMOIRE CENTRALE ET D'UN DISQUE A TETES FIXES

1- INTRODUCTION

1.1. POSITION DU PROBLEME

Dans ce chapitre nous allons proposer et analyser de nouvelles politiques pour gérer une configuration très typique de système à mémoire virtuelle. Le niveau rapide est constitué d'une mémoire centrale paginée à tores ou semi-conducteurs. Nous supposons qu'il n'existe pas de mémoire-tampon ou du moins que ce niveau supplémentaire, faisant partie de l'unité centrale, est extérieur au sous-système à optimiser. Le niveau lent est constitué d'un disque à secteurs ayant une tête de lecture/écriture par piste (ou d'un tambour, ce qui est équivalent du point de vue de l'adressage).



La portion d'une piste située à l'intérieur d'un secteur constitue l'unité de mémoire adressable et transférable sur ce type de support : nous supposerons que sa taille est égale à celle d'une page de la mémoire principale. Nous supposerons également qu'il existe une instruction permettant, par interrogation du contrôleur, de connaître instantanément la position angulaire du disque par rapport aux têtes de lecture et qu'en outre les commandes-canal peuvent être chaînées.

Nous considérerons le cas où chaque page a une image fixe sur le disque. Lorsqu'un défaut se produit la page remplacée est toujours recopiée en mémoire secondaire.

Nous avons déjà souligné l'influence néfaste des défauts de page. Des recherches pour limiter cet inconvénient ont été menées dans deux directions:

- minimisation du taux de défauts de page. Les ARP que nous avons présentés au chapitre deux ont été conçus dans ce but.

- minimisation du temps de réponse moyen du disque. Ce délai peut être réduit, en particulier, par l'utilisation de disques à têtes fixes, à secteurs

[1] [2]

Le but de l'étude présentée dans ce chapitre est d'analyser des stratégies globales d'allocation de la mémoire hiérarchisée qui concilient les exigences parfois contradictoires de la gestion de la mémoire centrale et de celle des mémoires secondaires rotatives.

Dans un système multiprogrammé, deux performances sont essentielles :

- La maximisation du taux d'activité de l'unité centrale c'est-à-dire de la fraction de temps η pendant laquelle l'unité centrale n'est pas oisive. Ce taux est directement lié au débit du système.

- La minimisation du temps de réponse pour chaque programme c'est-à-dire la minimisation de l'intervalle de temps réel qui sépare l'activation de ce programme de la fin de son exécution.

Limitons-nous momentanément au cas d'un système monoprogrammé à mémoire virtuelle paginée. Le critère η constitue alors une évaluation des deux performances évoquées ci-dessus. Il est égal à

$$\eta = \frac{e}{e + R}$$

où e est le temps moyen entre deux défauts de page consécutifs et R le temps moyen de suspension d'un programme à la suite d'un défaut de page.

Cette grandeur peut encore s'écrire :

$$\eta = \frac{1}{1 + FR}$$

où $F = \frac{1}{e}$ désigne le taux de défauts de page (évalué dans le "temps du programme"). La maximisation de η équivaut à la minimisation de FR . Les études de gestion de disque que nous avons citées ont pour but la minimisation de R , les études sur les ARP celle de F . Or ces optimisations peuvent reposer sur des exigences contradictoires. Il suffit pour s'en convaincre de considérer le cas suivant : supposons que l'image de la page sélectionnée par un ARP optimisé vienne juste de passer sous la tête de lecture-écriture du disque ; l'attente alors nécessaire contrebalance, et au-delà, le bénéfice de l'optimisation du remplacement. Les algorithmes que nous allons définir prennent leurs décisions non seulement en fonction de l'état de la

mémoire et du comportement du programme mais aussi, ce qui est nouveau, en fonction d'une prévision du temps de réponse du disque.

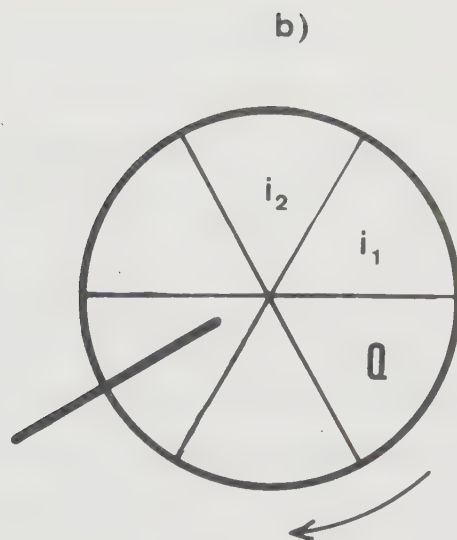
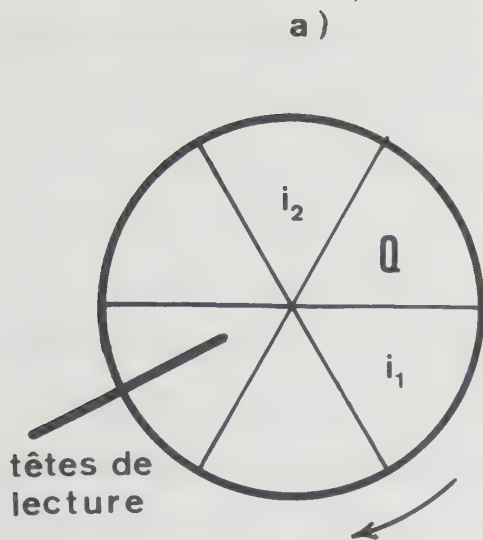
1.2. LES ALGORITHMES DE GESTION GLOBALE

Ces algorithmes sont des algorithmes de pagination à la demande. Désignons par m le nombre de pages physiques attribuées en mémoire centrale à un programme. Rappelons que les pages ont une image fixe sur le disque à secteurs et que toute page physique doit être recopiée avant de pouvoir être recouverte. Lorsqu'un défaut de page se produit, un algorithme de gestion globale d'ordre ℓ ($1 \leq \ell \leq m$) procède en deux temps pour déterminer la page à remplacer :

a) Tout d'abord, en fonction de l'état de la mémoire centrale et du comportement des programmes il détermine un ensemble $\mathbb{T}_\ell = \{i_1, i_2, \dots, i_\ell\}$ de pages susceptibles d'être déchargées. Par exemple $\mathbb{T}_\ell$ peut être l'ensemble des ℓ pages qui ont le plus long temps de résidence ininterrompue en mémoire (algorithme FIFO(ℓ)) ou l'ensemble des ℓ pages les plus anciennement référencées pour la dernière fois (algorithme LRU(ℓ)) ; les algorithmes "à préchargement partiel" maintiennent constamment en mémoire $(m-\ell)$ pages privilégiées : les autres pages figurant en mémoire centrale à l'instant d'un défaut de page constituent $\mathbb{T}_\ell$.

b) La page à décharger pour permettre l'entrée de la page Q faisant défaut est la page de $\mathbb{T}_\ell$ qui minimise le temps de réponse du disque. Si plusieurs pages minimisent ce temps, le choix entre elles est fait selon l'ordre de priorité de l'algorithme (FIFO ou LRU).

Le temps de réponse du disque et le choix de la page à décharger dépendent pour une part de ce qu'une page physique est - ou n'est pas -



La page à recopier est

avec tampon :

i_1

sans tampon :

i_1

avec tampon :

i_1

sans tampon :

i_1 ou i_2

(La page devant être transférée en mémoire centrale est Q)

FIGURE 1

réservée comme tampon en mémoire centrale. Si une telle réservation est faite, le chargement peut précéder le déchargement ce qui permettra éventuellement de réduire la durée de la suspension du programme : ceci est le cas par exemple sur la figure 1b (où $\ell = 2$).

Dans le cas d'un système multiprogrammé, l'algorithme est plus compliqué à mettre en oeuvre car il faut tenir compte de la longueur des files d'attente sur chaque secteur à l'instant du défaut de page. Les algorithmes - et en particulier leur adaptation au cas d'un système multiprogrammé avec partage dynamique de la mémoire - sont exposés plus en détail au paragraphe 3.1

Dans le paragraphe 2, nous analyserons des algorithmes de gestion globale à préchargement partiel de $(m-\ell)$ pages, dans le cadre d'un système monoprogrammé ; le comportement du programme sera supposé à références indépendantes. Au paragraphe trois nous présenterons une étude en simulation des algorithmes FIFO (ℓ) et LRU (ℓ), l'hypothèse de multiprogrammation étant alors retenue.

2 - ETUDE ANALYTIQUE DES ALGORITHMES DE GESTION GLOBALE A PRECHARGEMENT

PARTIEL .

Le système est monoprogrammé et ces algorithmes sont ainsi définis :

= les $(m-\ell)$ pages référencées avec la plus grande fréquence sont constamment présentes en mémoire centrale. Lors d'un défaut de page, l'ensemble Π_ℓ est constitué des ℓ autres pages qui se trouvent alors en mémoire centrale = la page de Π_ℓ qui est remplacée est celle dont le choix minimise le temps de réponse du disque. S'il y a plusieurs pages

qui se prêtent à cette minimisation, une d'elles est choisie au hasard.

S'il n'existe aucune corrélation entre la position de l'image de la page à charger et celle de la page à ranger, il est raisonnable de supposer que toutes les pages de Π_ℓ ont la même chance d'être sélectionnées.

Comme nous évaluons ces politiques sur des programmes conformes au modèle à références indépendantes, le taux de défauts de page est celui de l'algorithme $B_0(m-\ell, m)$ (voir chapitre 2, paragraphe 3.4.), lequel fait l'objet du théorème 2 du chapitre 2.

Nous allons ci-dessous calculer le temps de réponse du disque en distinguant selon qu'il existe ou non une page réservée comme tampon en mémoire principale. Puis nous procéderons à une étude numérique de l'influence sur l'efficacité du système du choix de ℓ . Détaillons les hypothèses faites sur l'utilisation de la mémoire secondaire.

Hypothèse 1 :

L'image physique d'une page sur le disque est fixe. Lorsqu'une page est déchargée de la mémoire centrale elle reprend sur le disque la place qu'elle y occupait avant son chargement en mémoire centrale.

Hypothèse 2 :

Le disque de pagination est à têtes fixes et caractérisé par :

S = nombre de secteurs

T = période de révolution

$a = \frac{T}{S}$ = durée de défilement d'un secteur

$\alpha = S^{-1}$

Le secteur de piste a une taille suffisante pour recevoir une page. La commutation lecture-écriture peut être effectuée dans l'intervalle de temps séparant le passage de la fin d'un secteur du passage du début du secteur suivant.

Hypothèse 3 :

Les positions sur le disque des pages p_i , $1 \leq i \leq l$, candidates au remplacement, sont indépendantes de la position sur le disque de la page requise.

Hypothèse 4 :

La répartition des pages du programme sur les différents secteurs du disque est uniforme. Posons

n_i = nombre de pages ayant leur image sur le secteur i .

La distribution de la variable $(n_1, n_2, \dots, n_s)$ est multinomiale, avec

$$\Pr [n_1 = x_1, \dots, n_s = x_s] = \frac{n!}{x_1! \dots x_s!} \frac{1}{s^n}$$

($n = \sum_{i=1}^s n_i$ = nombre de pages du programme).

Hypothèse 5 :

Le secteur sur lequel se trouve la tête de lecture/écriture à l'instant de la requête n'est accessible qu'après une révolution complète du disque. Au même instant, la position de la tête sur le secteur y est uniformément répartie.

Hypothèse 6 :

Tout défaut de page donne lieu à deux transferts :

- chargement en mémoire centrale de la page demandée
- recopie sur le disque de la page remplacée.

L'ordre dans lequel sont exécutées ces deux opérations dépend de l'existence d'une page - tampon en mémoire centrale.

Notations

Les notations des paragraphes précédents sont utilisées chaque fois qu'elles n'entraînent aucune ambiguïté. On pose de plus :

Q = page requise

L_j = nombre de secteurs séparant le secteur où se trouve la tête de lecture de celui où se trouve l'image de p_j , à l'instant de la requête ($j = 1 \dots \ell$)

D = [même définition que ci-dessus en remplaçant p_j par Q]

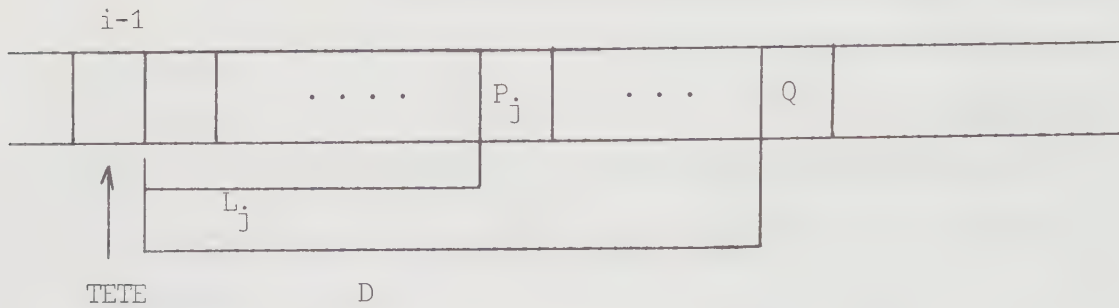
En outre, pour faciliter les calculs ultérieurs, les numéros de secteur sont à valeur dans les S classes d'entiers modulo S : i et $i + S$ désignent donc le même secteur.

2.1. CALCUL DU TEMPS MOYEN DE SERVICE SANS TAMPON R_{ST}

Nous étudions dans ce paragraphe le cas où aucune page n'est réservée en mémoire centrale comme tampon d'entrée/sortie : la recopie de la page remplacée doit donc être effectuée avant le chargement de la page Q .

Pour toute valeur d de D deux évènements exclusifs et exhaustifs sont à considérer :

$$i) \quad E_1(d) : (\exists j \in \{1, \dots, \ell\} \quad \text{tel que } L_j \leq d-1)$$

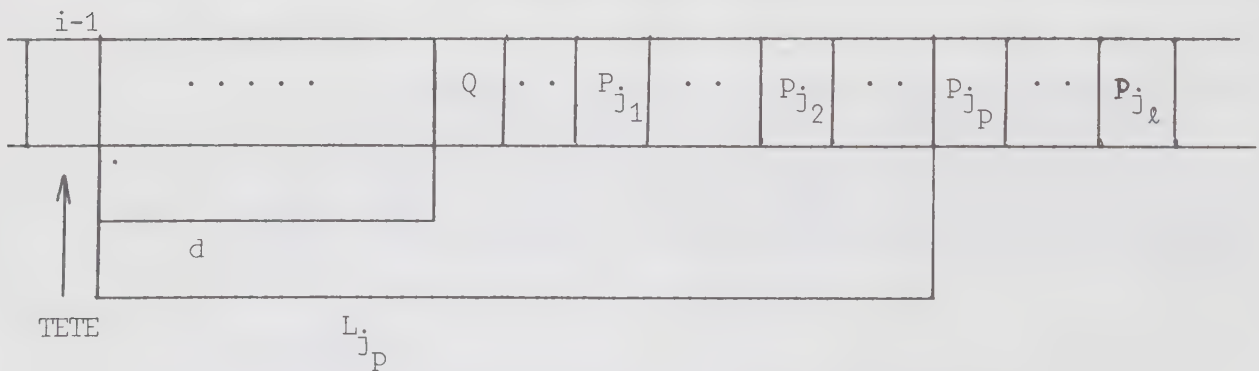


L'évènement $E_1(d)$, représenté sur la figure, est l'évènement favorable.

Le temps réponse du disque, compte-tenu du déplacement angulaire moyen des têtes sur le secteur où elles se trouvent à l'instant de la requête (voir l'hypothèse 5) est :

$$t_1(d) = \frac{2d + 3}{2} a$$

$$ii) \quad E_2(d) : (\forall j \in \{1, \dots, \ell\} : L_j \geq d)$$



Le temps de service moyen est alors :

$$t_2(d) = \frac{2 S + 2 d + 3}{2} a$$

Calcul des probabilités des évènements $E_1(d)$ et $E_2(d)$

Ces deux évènements étant exclusifs l'un de l'autre,

$$\Pr \{E_1(d)\} + \Pr \{E_2(d)\} = \Pr \{E_1(d) \cup E_2(d)\} = \Pr \{D=d\}$$

et l'on obtient :

$$\begin{aligned} \Pr \{D=d\} &= \sum_{i=1}^S \Pr \{i\} \Pr \{d|i\} \\ &= \sum_{i=1}^S \alpha \frac{n_{i+d}}{n} = \alpha \end{aligned}$$

Il suffit donc de calculer $\Pr \{E_2(d)\}$:

$$\Pr \{E_2(d)\} = \sum_{i=1}^S \Pr \{A\} \times \Pr \{B\}$$

où l'on a posé :

$A = \left[\text{la tête de lecture se trouve sur le secteur } i-1 \text{ lors de la requête} \right]$

$B = \left[\text{l'image de } Q \text{ est sur } i+d ; \text{ les pages } p_j \text{ sur } i+d, \text{ ou } i+d+1, \dots \right. \\ \left. \dots, \text{ ou } i+S-1 \right]$

Explicitons $\Pr \{B\}$.

$\Pr \{B\} = \Pr \{ \text{image de } Q \text{ sur } i+d \}$

$\times \Pr \{ p_1 \text{ sur } i+d \text{ ou } i+d+1 \text{ ou } \dots \text{ ou } i+S-1 \mid Q \text{ sur } i+d \}$

$\times \Pr \{ p_j \text{ sur } i+d \text{ ou } i+d+1 \text{ ou } \dots \text{ ou } i+S-1 \mid Q \text{ sur } i+d ;$

$$\begin{aligned}
 & P_1, P_2, \dots, P_{j-1} \text{ sur } i+d \text{ ou } i+d+1 \dots \text{ou } i+S-1) \\
 & \dots\dots\dots \\
 & \times \Pr (p_\ell \text{ sur } i+d \text{ ou } i+d+1 \text{ ou } \dots \text{ ou } i+S-1 \mid Q \text{ sur } i+d ; \\
 & \quad P_1, P_2, \dots, P_{\ell-1} \text{ sur } i+d \text{ ou } i+d+1 \dots \text{ou } i+S-1) \\
 \\
 \Pr (B) &= \frac{n_{i+d}}{n} \times \frac{n_{i+d} + n_{i+d-1} + \dots + n_{i+S-1} - 1}{n-1} \\
 & \dots\dots\dots \\
 & \times \frac{n_{i+d} + n_{i+d-1} + \dots + n_{i+S-1} - j}{n-j} \\
 & \dots\dots\dots \\
 & \times \frac{n_{i+d} + n_{i+d-1} + \dots + n_{i+S-1} - \ell}{n-\ell}
 \end{aligned}$$

On en déduit, en remarquant que $\Pr (A) = \alpha$

$$\begin{aligned}
 \Pr (E_2(d) \mid n_1, \dots, n_S) &= \sum_{i=1}^S \alpha \frac{n_{i+d}}{n} e_i \\
 \\
 \text{où } e_i &= \left\{ \begin{array}{ll} 0 & \text{si } \sum_{r=d}^{S-1} n_{i+r} < \ell+1 \\ \frac{(\sum_{r=d}^{S-1} n_{i+r} - 1)! (n-\ell-1)!}{(\sum_{r=d}^{S-1} n_{i+r} - \ell-1)! (n-1)!} & \text{sinon} \end{array} \right.
 \end{aligned}$$

En utilisant l'hypothèse 4, nous pouvons écrire :

$$\Pr (E_2(d)) = \sum_{i=1}^S \alpha \frac{n_{i+d}}{n} \sum_{r=d}^{S-1} \frac{(\sum_{r=d}^{S-1} n_{i+r} - 1)! (n-\ell-1)!}{(\sum_{r=d}^{S-1} n_{i+r} - \ell-1)! (n-1)!} \times \frac{n!}{n_1! n_2! \dots n_S!} \times \frac{1}{S^n}$$

la sommation $\sum_{*}$ étant étendue à tous les S-uples $(n_1, n_2, \dots, n_S)$ d'entiers non-négatifs tels que

$$\sum_{i=1}^S n_i = n,$$

les entiers $n_{i+d}, n_{i+d+1}, \dots, n_{i+S-1}$ vérifiant en outre

$$\sum_{r=d}^{S-1} n_{i+r} \geq l+1$$

En modifiant l'arrangement des termes, $\text{Pr } (E_2(d))$ devient :

$$\text{Pr } (E_2(d)) = \alpha \sum_{i=1}^S \frac{(n-l-1)!}{S^n n!} \sum_{*} n_{i+d} \frac{\left(\sum_{r=d}^{S-1} n_{i+r} - 1 \right)!}{\left(\sum_{r=d}^{S-1} n_{i+r} - l - 1 \right)!} \frac{n!}{n_1! n_2! \dots n_S!}$$

Le lemme ci-dessous, démontré en annexe, permet d'évaluer l'expression précédente.

Lemme :

Soit $(y_1, y_2, \dots, y_p, y_{p+1}, \dots, y_S)$ une permutation de $(1, 2, \dots, S)$.

Soit E , fonction de n, l, S, p et $(y_1, y_2, \dots, y_p)$, la quantité :

$$\sum_{i=1}^S \frac{(n-l-1)!}{S^n n!} \sum_{*} n_{i+y_1} \frac{(n_{i+y_1} + n_{i+y_2} + \dots + n_{i+y_p} - 1)!}{(n_{i+y_1} + \dots + n_{i+y_p} - l - 1)!} \frac{n!}{n_1! n_2! \dots n_S!}$$

la sommation $\sum_{*}$ étant étendue aux S-uples d'entiers non-négatifs $(n_1, n_2, \dots, n_S)$ tels que

$$\sum_{i=1}^S n_i = n,$$

les entiers non-négatifs $n_{i+y_1}, \dots, n_{i+y_p}$ vérifiant en outre

$$\sum_{j=1}^p n_{i+y_j} \geq l + 1$$

Alors

$$E = \left(\frac{p}{S} \right)^l$$

En utilisant ce lemme (avec $p = S-d$), nous obtenons l'expression suivante de $\text{Pr } (E_2(d))$:

$$\text{Pr } (E_2(d)) = \alpha \left(\frac{S-d}{S} \right)^l = \alpha (1-\alpha d)^l$$

et par suite :

$$\text{Pr } (E_1(d)) = \alpha - \text{Pr } (E_2(d)) = \alpha (1-(1-\alpha d)^l)$$

Temps de réponse moyen du disque (sans tampon)

$$R_{ST} = \sum_{d=0}^{S-1} [t_1(d) \times \text{Pr } (E_1(d)) + t_2(d) \text{Pr } (E_2(d))]$$

$$R_{ST} = \sum_{d=0}^{S-1} \left[\frac{2d+3}{2} \alpha (1-(1-d\alpha)^l) + \frac{2S+2d+3}{2} \alpha (1-d\alpha)^l \right]$$

et finalement

$$R_{ST} = T \left(\frac{S+4}{2S} + \frac{1}{S} \sum_{i=1}^{S-1} \left(\frac{i}{S} \right)^l \right)$$

Remarques

Le résultat obtenu est indépendant de n , taille du programme considéré. Les conclusions sont donc indépendantes des caractéristiques des programmes des utilisateurs du système, et ne dépendent que de ses données techniques.

Lorsque l tend vers l'infini, R_{ST} tend vers la limite $T \cdot \frac{S+4}{2S}$. Pour $S=3$, cette limite est atteinte à 1% près dès que l est supérieur à 8, à 10% près dès que l est supérieur à 2.

2.2. CALCUL DU TEMPS MOYEN DE SERVICE AVEC TAMPON R_{AT}

Dans ce cas, à chaque défaut de page, on choisit celle des deux méthodes suivantes qui minimise le temps de service.

i) Méthode sortie-entrée.

Parmi les l pages candidates au remplacement, on recopie sur le disque celle dont l'image passe la première sous la tête. On charge ensuite la page requise dans l'emplacement libéré.

ii) Méthode entrée-sortie.

La page requise est chargée dans le tampon. On recopie ensuite sur le disque une des pages candidates. L'emplacement libéré devient le nouveau tampon.

On est amené à considérer le système exhaustif d'évènements deux à deux exclusifs, représenté dans le tableau ci-dessous :

	DEFINITION	METHODE CHOISIE	DUREE DU SERVICE
$E_0(d)$	$\exists j \in \{1, \dots, l\} : L_j \leq d-1$	S-E	$\frac{2d+3}{2} a$
$E_\beta(d)$ (β est un entier: $1 \leq \beta \leq S-d-1$)	$\exists j \in \{1, \dots, l\} : L_j = d+\beta$ <u>ET</u> $\forall j \in \{1, \dots, l\} : \begin{cases} L_j \geq d+\beta \\ \text{OU} \\ L_j = d \end{cases}$	E-S	$\frac{3+2(d+\beta)}{2} a$
$E^*(d)$	$\forall j \in \{1, \dots, l\} : L_j = d$	Indifférente	$\frac{2d+3+2S}{2} a$

Pour calculer la probabilité de l'évènement $E_\beta(d)$, remarquons que, d étant donné, l'évènement complémentaire de $A_i^{\beta+1}(d)$, $1 \leq \beta \leq S-d-1$, s'énonce ainsi :

"Il existe au moins une page p_j dont l'image n'est ni sur le secteur $i+d$, ni sur aucun des secteurs numérotés $i+\gamma$, avec $d+\beta+1 \leq \gamma \leq S-1$ ".

Notons $\neg X$ l'évènement complémentaire de l'évènement X ; on en déduit, d étant donné,

$$A_i^\beta(d) \wedge \neg A_i^{\beta+1}(d) = \begin{cases} \exists j \in \{1, \dots, l\} & : L_j = d + \beta \\ \text{ET} \\ \forall j \in \{1, \dots, l\} & : \begin{cases} L_j \geq d + \beta \\ \text{OU} \\ L_j = d \end{cases} \end{cases}$$

Par conséquent

$$\Pr(E_\beta(d)) = \sum_{i=1}^S \Pr(A) \times \Pr(C)$$

avec A défini comme précédemment

et $C = (A_i^\beta(d) \wedge \neg A_i^{\beta+1}(d) \quad \text{ET} \quad D=d)$

Explicitons $\Pr(C)$:

$$\Pr(C) = \Pr(\text{page } Q \text{ sur } i+d) \times \Pr(A_i^\beta(d) \wedge \neg A_i^{\beta+1}(d) \mid D=d)$$

Nous pouvons écrire par ailleurs :

$$\Pr(A_i^\beta(d) \wedge \neg A_i^{\beta+1}(d) \mid D=d) + \Pr(A_i^\beta(d) \wedge A_i^{\beta+1}(d) \mid D=d) = \Pr(A_i^\beta(d) \mid D=d)$$

et remarquant que $A_i^\beta(d) \wedge A_i^{\beta+1}(d) = A_i^{\beta+1}(d)$, nous obtenons :

$$\Pr(A_i^\beta(d) \wedge \neg A_i^{\beta+1}(d) \mid D=d) = \Pr(A_i^\beta(d) \mid D=d) - \Pr(A_i^{\beta+1}(d) \mid D=d)$$

et donc :

$$\Pr \{E_{\beta}(d)\} = \sum_{i=1}^S \alpha \frac{n_{i+d}}{n} (\Pr \{A_i^{\beta}(d) \mid D=d\} - \Pr \{A_i^{\beta+1}(d) \mid D=d\})$$

Désignons par $T(\beta)$ la somme

$$T(\beta) = \sum_{i=1}^S \alpha \frac{n_{i+d}}{n} \Pr \{A_i^{\beta}(d) \mid D=d\}$$

L'égalité devient

$$\Pr \{E_{\beta}(d)\} = T(\beta) - T(\beta+1)$$

Posons

$$\sigma = n_{i+d} + n_{i+d+\beta} + n_{i+d+\beta+1} + \dots + n_{i+S-1}$$

Par un calcul analogue à celui de $\Pr [B]$ au paragraphe 2.1 nous obtenons :

$$\Pr \{A_i^{\beta}(d) \mid D=d ; n_1, n_2, \dots, n_S\} = \begin{cases} 0 & \text{si } \sigma < \ell + 1 \\ \frac{(\sigma-1)! (n-\ell-1)!}{(\sigma-\ell-1)! (n-1)!} & \text{si } \sigma \geq \ell + 1 \end{cases}$$

En utilisant de nouveau l'hypothèse 4, nous pouvons écrire :

$$\Pr \{A_i^{\beta}(d) \mid D=d\} = \sum_{*} \frac{(\sigma-1)! (n-\ell-1)!}{(\sigma-\ell-1)! (n-1)!} \times \frac{n!}{n_1! n_2! \dots n_S!} \times \frac{1}{S^n}$$

la sommation $\sum_{*}$ étant étendue aux S -uples d'entiers non-négatifs $(n_1, n_2, \dots, n_S)$ vérifiant :

$$n_{i+d} + n_{i+d+\beta+1} + \dots + n_{i+S-1} \geq \ell + 1$$

;

$$n_1 + n_2 + \dots + n_S = n$$

$T_{(\beta)}$ s'écrit donc :

$$T_{(\beta)} = \alpha \sum_{i=1}^S \frac{n_i}{n} \sum_{*} \frac{(\sigma-1)! (n-\ell-1)!}{(\sigma-\ell-1)! (n-1)!} \times \frac{n!}{n_1! n_2! \dots n_S!} \times \frac{1}{S^n}$$

$$T_{(\beta)} = \alpha \sum_{i=1}^S \frac{(n-\ell-1)!}{S^n n!} \times \sum_{*} \frac{(\sigma-1)!}{(\sigma-\ell-1)!} \times \frac{n!}{n_1! n_2! \dots n_S!}$$

Nous utilisons le lemme avec dans ce cas $y_1 = d$, $y_2 = d+\beta$, $y_3 = d+\beta+1, \dots, y_p = S-1$, soit $p=S-d-\beta+1$. D'où

$$T_{(\beta)} = \alpha \left(\frac{S-d-\beta+1}{S} \right)^\ell = \alpha^{\ell+1} (S-(d+\beta)+1)^\ell$$

et par suite :

$$\Pr (E_\beta(d)) = \alpha^{\ell+1} \{ (S-(d+\beta)+1)^\ell - (S-(d+\beta))^\ell \}$$

Evènement $E^*(d)$

Considérons l'évènement $A_i^{S-d}(d)$. D'après la définition de $A_i^\beta(d)$, les images des pages p_j , $1 \leq j \leq \ell$, sont toutes sur le secteur $i+d$. On en déduit, comme précédemment :

$$\Pr (E^*(d)) = \sum_{i=1}^S \Pr (A) \times \Pr (A_i^{S-d}(d) | D=d)$$

et par définition de $T_{(\beta)}$

$$\Pr (E^*(d)) = T(S-d)$$

soit

$$\Pr (E^*(d)) = \alpha^{\ell+1}$$

Temps de réponse moyen du disque (avec tampon)

Le temps de réponse moyen R_{AT} est obtenu par les trois sommations correspondant aux trois familles d'évènements, suivant les valeurs de d .

Evènements $E_0(d)$

$$R_1 = a \alpha \sum_{d=0}^{S-1} \frac{2d+3}{2} (1-(1-d\alpha))^l$$

Evènements $E_\beta(d)$

Remarquons en premier lieu que les résultats, pour cette famille d'évènements, ne dépendent que de $\beta+d$. On en déduit, $\beta+d$ variant entre 1 et $S-1$,

$$R_2 = \alpha^{l+1} \sum_{t=1}^{S-1} t \frac{3+2t}{2} ((S-t+1)^l - (S-t)^l)$$

Evènements $E^*(d)$

$$R_3 = a \alpha^{l+1} \frac{3S^2 + 2S}{2}$$

Puisque $R_{AT} = R_1 + R_2 + R_3$, nous obtenons finalement

$$R_{AT} = a \alpha \sum_{d=0}^{S-1} \frac{2d+3}{2} [1-(1-d\alpha)^l] + \alpha^{l+1} \sum_{t=1}^{S-1} t \frac{3+2t}{2} [(S-t+1)^l - (S-t)^l] + \alpha^{l+1} \frac{3S^2+2S}{2}$$

Remarque

Lorsque l tend vers l'infini, R_{AT} tend vers la limite $T \frac{S^2+2S+2}{2S^2}$.

Cette limite est atteinte à 1% près pour $l > 8$, à 10% près pour $l > 2$ lorsque $S = 3$

1.3. RESULTATS NUMERIQUES

Soient m le nombre total de pages physiques allouées au programme et m' le nombre de pages physiques destinées à l'exécution proprement dite ; m et m' sont égaux s'il n'est pas réservé de page tampon ; sinon $m' = m-1$. Les algorithmes de remplacement de page que nous étudions maintiennent constamment en mémoire les k pages ($0 \leq k \leq m'-1$) ayant les plus fortes probabilités d'être référencées. Lors d'un défaut de page, ils choisissent la page à remplacer parmi les ℓ ($\ell = m'-k$) autres pages présentes en mémoire de telle façon que le temps de service du disque soit minimum. Comme nous l'avons expliqué au début du paragraphe 2, il est raisonnable d'admettre que chacune de ces pages a la probabilité $1/\ell$ d'être désignée. Le taux de défauts de page des algorithmes que nous considérons est donc celui des ARPAPP optimaux.

L'efficacité du système de pagination est

$$\eta_{ST}(\ell) = \frac{1}{1+F(B_0(m-\ell, m)) \times R_{ST}(\ell)} \quad 1 \leq \ell \leq m$$

si aucune page n'est réservée comme tampon et :

$$\eta_{AT}(\ell) = \frac{1}{1+F(B_0(m-1-\ell, m-1)) \times R_{AT}(\ell)} \quad 1 \leq \ell \leq m-1$$

dans le cas contraire.

2.3.1. DETERMINATION DE LA VALEUR OPTIMALE DE n

Sur la figure 2 est représentée la variation en fonction de ℓ du temps moyen de service d'un disque à trois secteurs. L'amélioration apportée

par la possibilité de choisir la page à ranger croît rapidement pour les premières valeurs de ℓ puis devient presque constante.

La figure 3 montre la variation du taux de défauts de page en fonction de ℓ , la taille m' de la zone de mémoire réservée à l'exécution du programme étant constante ($m'=13$). Trois types de programmes sont considérés qui diffèrent entre eux par la distribution $(\beta_i)_{i=1,\dots,n}$ des probabilités de référence :

- pour le programme à distribution uniforme des références, toutes les probabilités β_i sont égales à $1/n$.

- pour le programme GEOM (0,95), la distribution (β_i) est géométrique tronquée :

$$\beta_i = C_1 \times (0,95)^i \quad \text{pour } i=1,\dots,n$$

où

$$C_1 = \frac{0,05}{0,95(1-(0,95)^{n+1})} = 0,08204 \quad (\text{constante de normalisation})$$

- pour le programme 99992222111111111111, la distribution des probabilités de référence est telle que :

$$\beta_i = C_2 \times 9 \quad \text{pour } 1 \leq i \leq 4$$

$$\beta_i = C_2 \times 2 \quad \text{pour } 5 \leq i \leq 8$$

$$\beta_i = C_2 \times 1 \quad \text{pour } 9 \leq i \leq 20$$

où C_2 désigne la constante de normalisation.

On constate, ce qui n'est pas étonnant, que ces courbes sont croissantes. Vantilborgh [4] a d'ailleurs montré que pour $0 \leq k \leq m'-2$,

$$0 \leq F(B_0(k,m')) - F(B_0(k+1,m')) < \frac{1}{m'-k} \left(1 - \sum_{i=1}^n \beta_i^2\right)$$

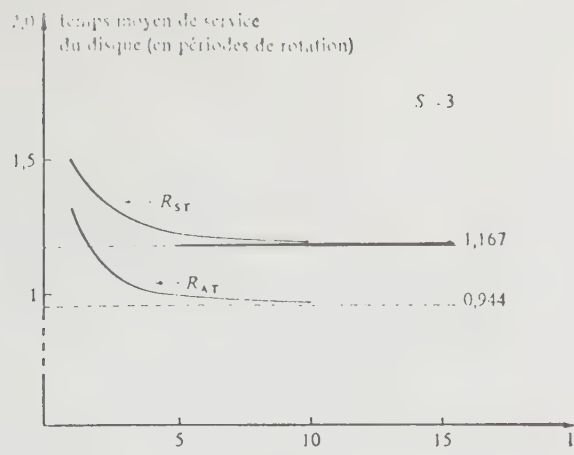


Fig. 2

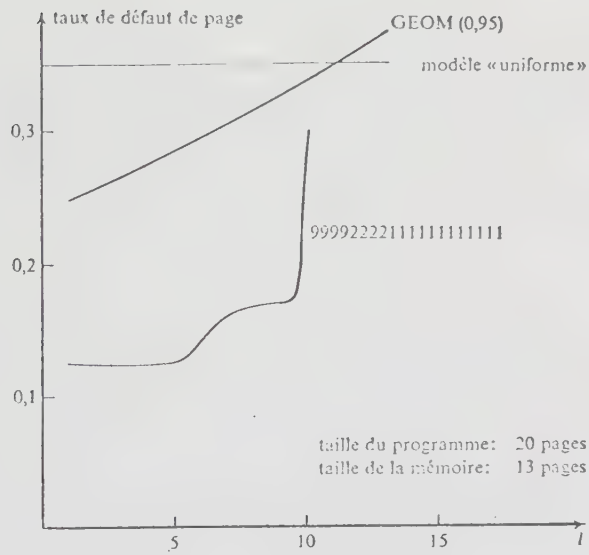


Fig. 3

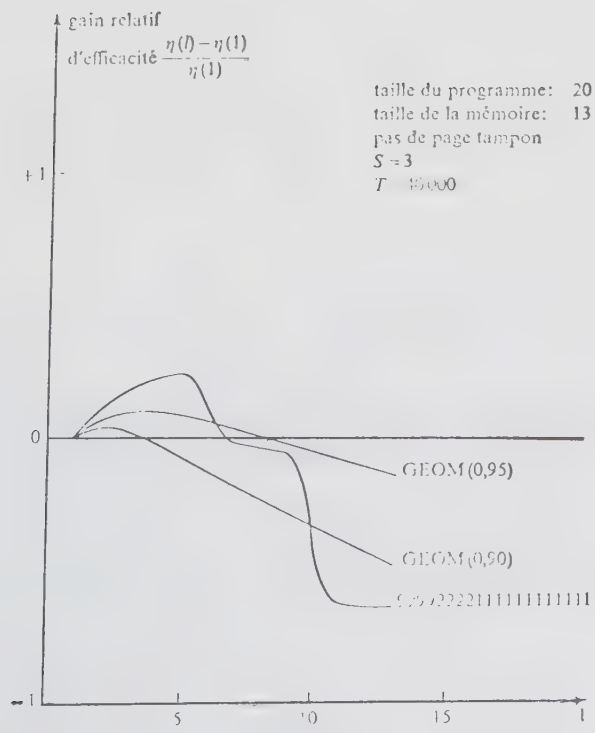


Fig. 4

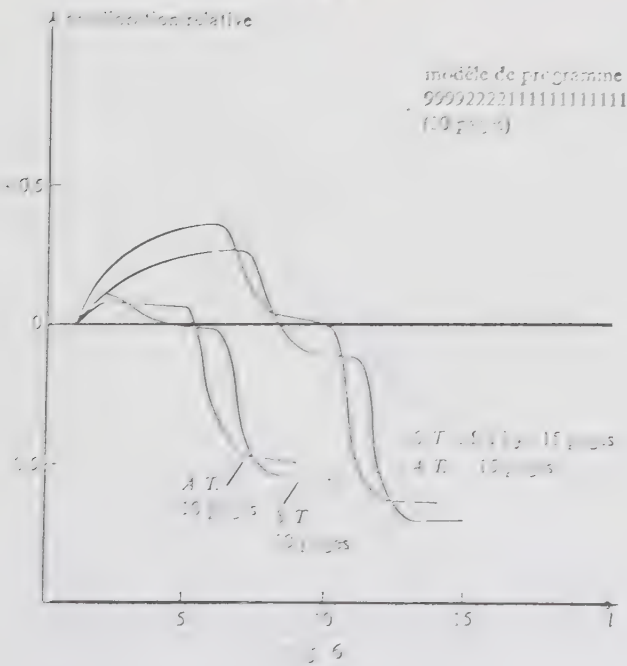
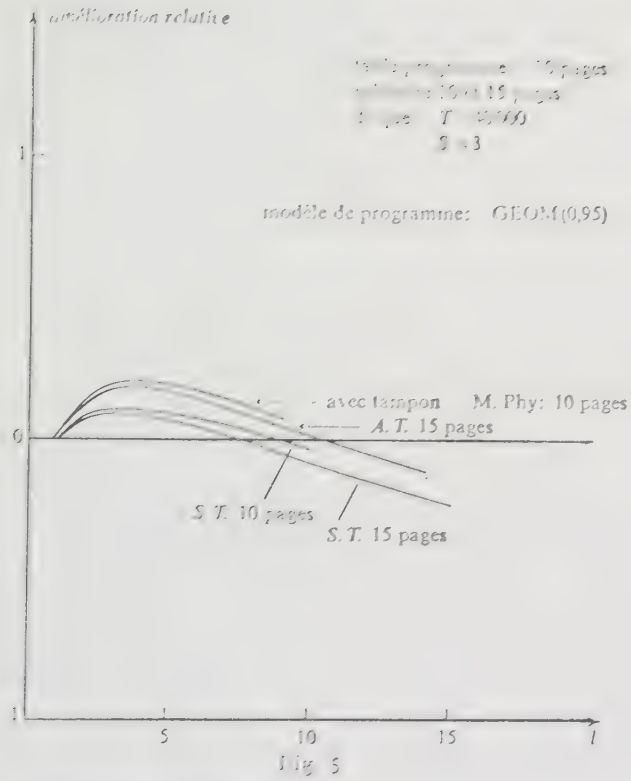
La courbe représentant le taux de défauts de page du troisième programme a deux zones de croissance rapide : pour $\ell = 6$ (resp. 10) une page dont la probabilité d'être référencée est proportionnelle à 2 (resp. 9) disparaît de l'ensemble ϕ_k des pages préchargées.

Considérant ces courbes représentant les variations du temps de réponse du disque et du taux de défauts de page, on peut prévoir que les algorithmes que nous étudions seront efficaces pour de petites valeurs de ℓ . C'est ce que confirment les courbes des figures 4, 5 et 6 qui représentent en fonction de ℓ l'amélioration relative d'efficacité :

$$\frac{\eta(\ell) - \eta(1)}{\eta(1)}$$

L'amélioration relative peut être importante : plus de 23% pour le programme 99992222111111111111 lorsque $\ell = 4$ (figure 4) ; 36% pour ce même programme lorsque $\ell = 6$, une page étant réservée comme tampon (figure 6). L'amélioration relative est généralement plus importante lorsqu'on utilise un tampon ; ceci ne nous livre néanmoins aucun renseignement sur l'utilité de ce tampon puisque les efficacités η_{AT} (resp. η_{ST}) sont comparées entre elles.

Faisant varier la distribution $(\beta_i)_{i=1, \dots, n}$, la taille du programme et celle de la mémoire ainsi que le nombre de secteurs et la période de rotation des disques, nous avons obtenu quelques centaines de courbes de ce type. Toutes ont un seul maximum. Celui-ci est très souvent atteint pour une valeur de ℓ inférieure ou égale à 5. L'efficacité est proche de



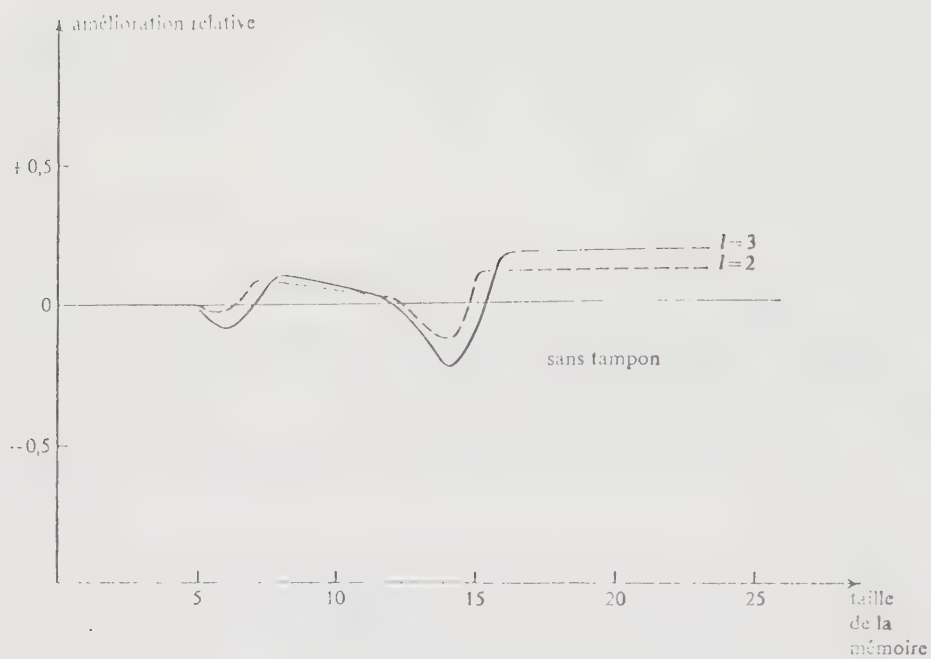


Fig. 7

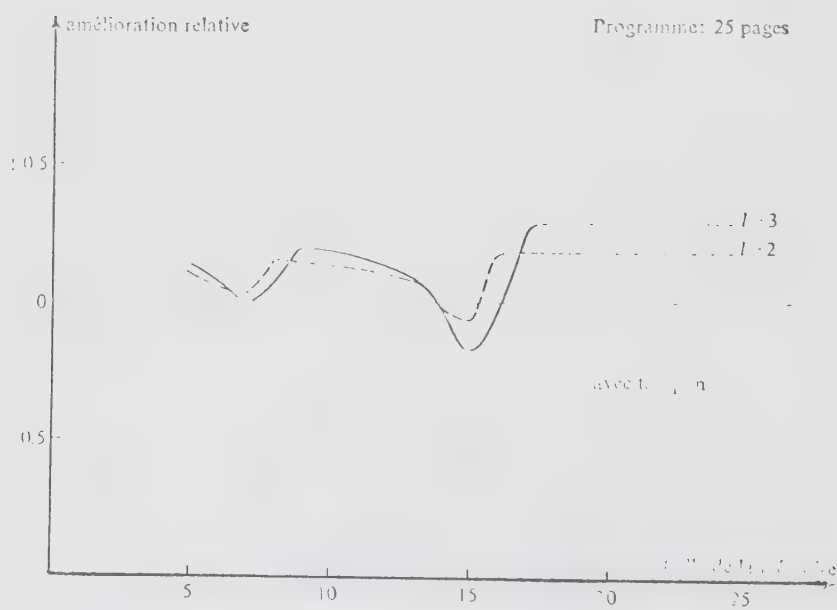
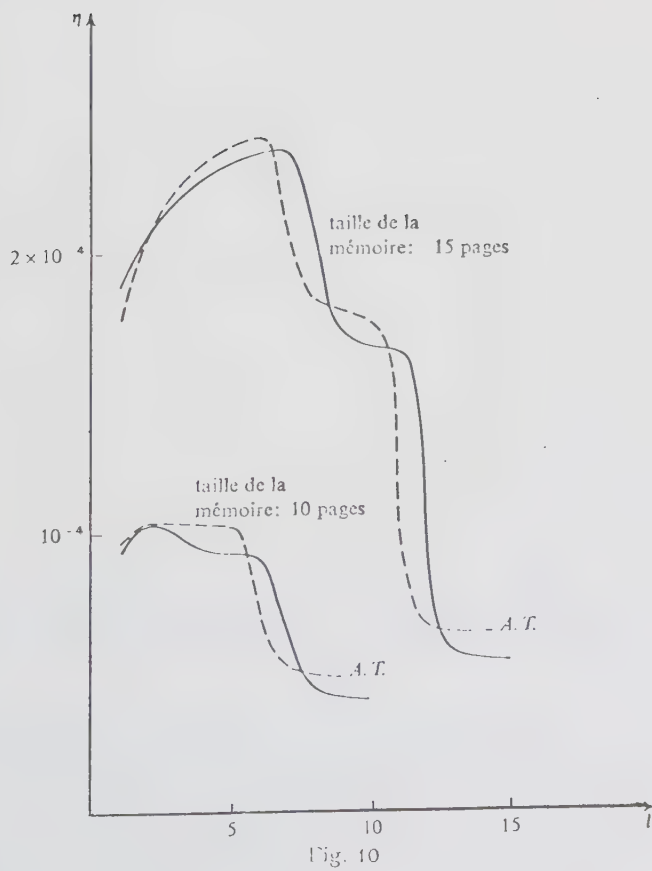
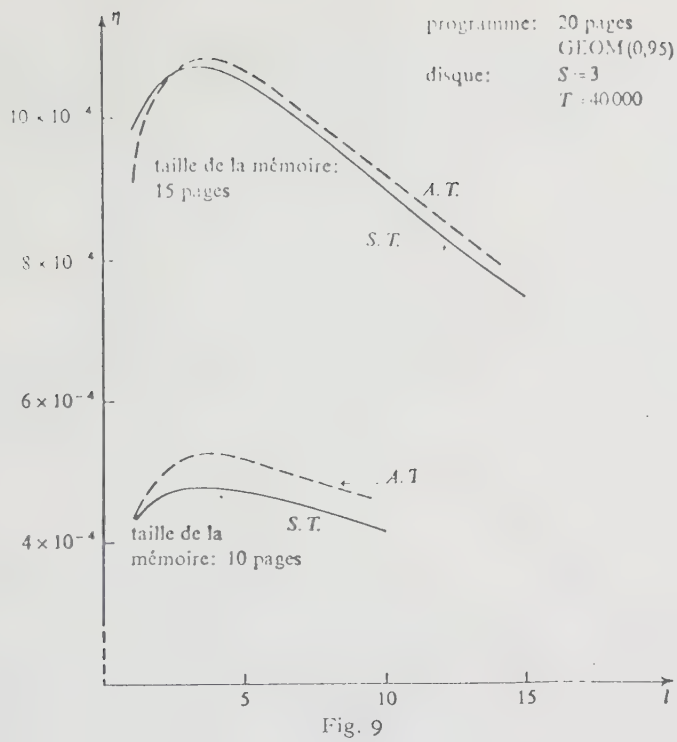


Fig. 8



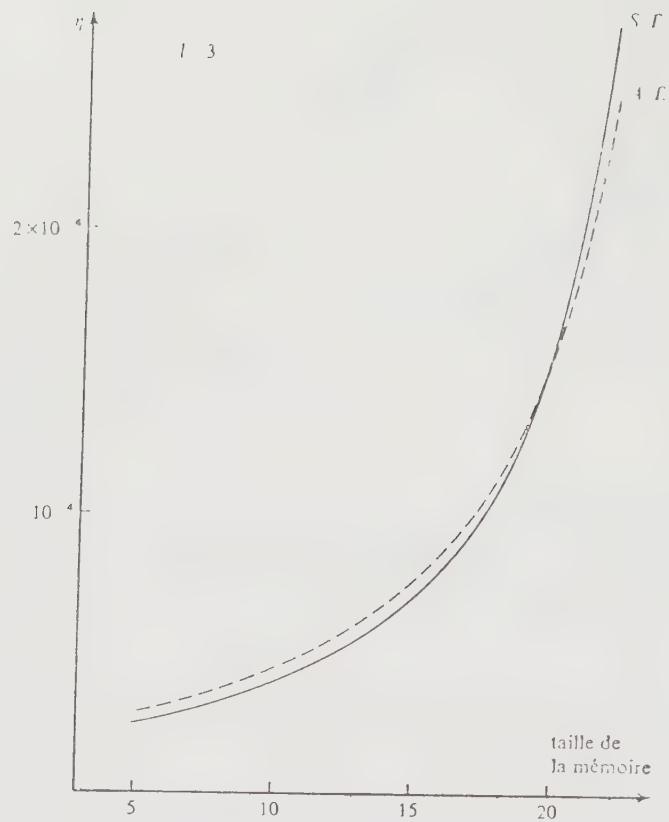


Fig. 11

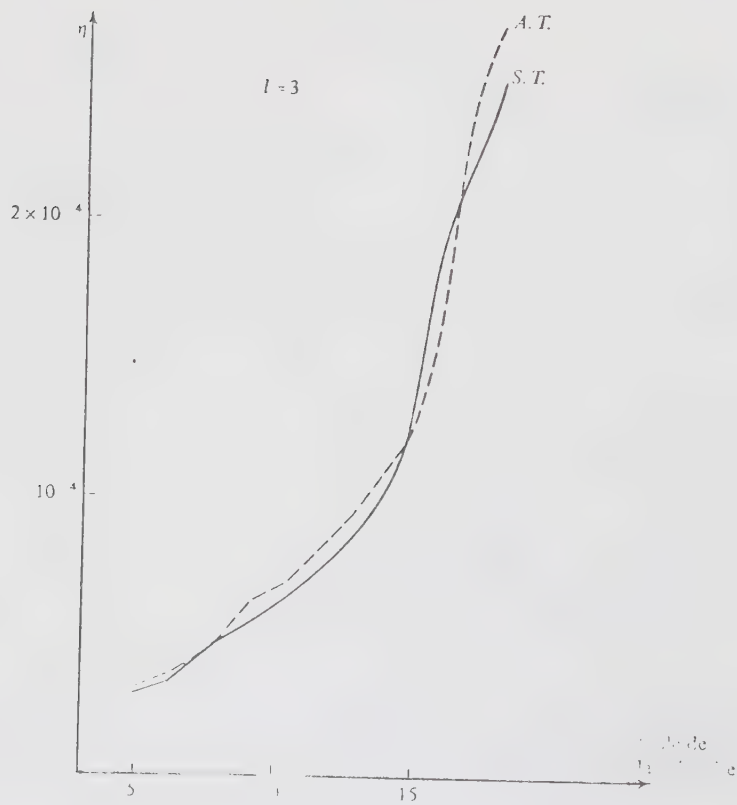


Fig. 12

sa valeur maximum pour $\ell = 2$ ou 3 . En outre pour ces deux valeurs, il est rare que les performances soient moins bonnes que pour $\ell = 1$. Cela arrive pourtant (cf. figures 7 et 8).

La variation de l'amélioration relative en fonction de la taille de la mémoire est représentée pour $\ell = 2$ et $\ell = 3$ sur les figures 7 (sans tampon) et 8 (avec tampon).

2.3.2. UTILITE DU TAMPON

De la considération d'un grand nombre de situations correspondant à divers types de programme et différentes tailles de mémoire, il ressort que le maximum atteint par $\eta_{AT}(\ell)$ est presque toujours supérieur à celui de $\eta_{ST}(\ell)$, les valeurs optimales de ℓ n'étant pas nécessairement les mêmes dans les deux cas. Cela apparaît par exemple sur les figures 9 et 10.

Il apparaît toutefois difficile de tirer de ces résultats numériques des remarques suffisamment générales. Adoptons par exemple un point de vue pratique et choisissons pour ℓ une valeur convenable (1, 2 ou 3). On peut alors tracer de courbes analogues à celles des figures 11 et 12 : elles ne permettent pas de trancher le problème de l'utilité d'un tampon. Tout au plus peut-on remarquer que si la taille de la mémoire est proche de la taille du programme, la réservation d'un tampon est néfaste (à la limite, lorsque $m=n$, il n'y a plus de défauts de page que dans le cas où l'on garde une page physique pour loger le tampon).

3 - ETUDE EN SIMULATION DES ALGORITHMES FIFO (ℓ) ET LRU (ℓ)

L'étude analytique du paragraphe 2 concerne les algorithmes de gestion globale à préchargement partiel. Nous allons l'étendre, par simulation, au cas des algorithmes FIFO(ℓ) et LRU(ℓ) ; nous considérons aussi bien les systèmes monoprogrammés que les systèmes multiprogrammés. Dans le paragraphe 3.1. nous allons décrire brièvement le simulateur employé. A cette occasion les algorithmes retenus pour les systèmes multiprogrammés seront exposés en détail.

3.1. STRUCTURE ET CONCEPTION DU SIMULATEUR

Le simulateur utilisé dans cette étude a été divisé en plusieurs modules qui représentent chacun une des unités fonctionnelles du système :

- le module "comportement de programmes" qui produit la suite des références aux pages générée par chaque programme en cours d'exécution.
- le module "mémoire centrale" qui retient et gère l'état de la mémoire centrale : ensemble de pages appartenant à chaque programme, listes de priorité de l'algorithme de remplacement.
- le module "mémoire secondaire" qui gère les files d'attente du disque à secteurs et simule les transferts.
- le module "superviseur" qui, outre la synchronisation assure l'activation des programmes et le prélèvement des mesures.
- le module "algorithme de remplacement de pages" qui choisit la page à remplacer en fonction des informations fournies par tous les modules précédents et qui à son tour leur communique sa décision.

Cette décomposition du simulateur en modules permet de modifier aisément les modèles et les algorithmes employés. Une description détaillée figure dans [5].

3.1.1. LE MODULE "COMPORTEMENT DES PROGRAMMES"

Le modèle de comportement retenu pour cette étude est le modèle à liste LRU, à distances indépendantes (cf. chapitre 1, paragraphe 2.4) Les probabilités (α_i) peuvent être évaluées facilement à partir de vrais programmes ; en outre l'accord entre ce modèle et la réalité du comportement est bon, du moins tant que l'on ne considère que des propriétés locales.

Pour chaque programme, deux listes sont réservées dans la zone de données du module. La première est celle des pages classées suivant l'ancienneté de leur dernière référence. La deuxième contient les probabilités α_d de référencer la $d^{\text{ème}}$ page de la liste précédente. La taille n de chaque programme, ainsi que la distribution $(\alpha_d)_{d=1,..,n}$, seront déclarées par l'utilisateur du simulateur. Pour le programme en cours d'exécution les numéros des pages référencées sont obtenus d'un générateur de nombres pseudo-aléatoires selon la méthode classique de transformation d'un tirage congruentiel uniforme entre 0 et 1. Après chaque référence le contrôle est passé au module "modèle de mémoire centrale".

Remarquons que si l'on souhaite changer de modèle de comportement ou utiliser des chaînes de références relevées pendant l'exécution d'un programme réel, il suffit de modifier ce module.

3.1.2. LE MODULE "MEMOIRE CENTRALE"

Ce module gère l'état de la mémoire centrale qui est défini à chaque instant par les données suivantes :

- le nombre de pages physiques accordées à chaque programme. Il est fonction des références générées, de l'algorithme de remplacement et de

l'ordonnancement des programmes par le superviseur.

- l'identité des pages présentes en mémoire centrale, c'est-à-dire les couples (numéro de page virtuelle, numéro de programme).

- l'ensemble des pages déchargeables défini par le module "algorithme de remplacement de pages".

Lorsque le contrôle est reçu du module décrit dans le paragraphe 3.1.1., la présence de la page référencée est testée dans les données du module. En cas de succès le contrôle est rendu au module précédent. Au cas où un défaut de page est constaté, le contrôle est passé au module "algorithme de remplacement de pages".

3.1.3. LE MODULE "MEMOIRE SECONDAIRE"

Les caractéristiques physiques de la mémoire secondaire simulée sont celles d'un disque à têtes fixes et à secteurs. On représente le nombre de secteurs et le temps de révolution du disque ainsi que la position du bras de lecture par rapport à sa surface. On suppose que le disque est suffisamment grand pour que toutes les pages des programmes puissent avoir une image à une adresse (numéro de piste, numéro de secteur) fixe sur disque.

Pour chaque programme, le module contiendra une table des emplacements de ses pages sur le disque. Le système de gestion du disque contient une file d'attente par secteur pour réaliser la politique dite d'Eschenbach [4]. La table contenant la longueur de la file d'attente sur chaque secteur ainsi que la variable contenant la position du bras par rapport à la surface du disque sont maintenues par ce module et consultées par le module "algorithme de remplacement de pages".

A chaque transfert de page décidé, le module utilise une horloge interne pour calculer la date à laquelle l'état de la mémoire centrale sera modifié : cette date est celle de la fin du transfert pour un chargement et de son début pour un déchargement. Chaque fois qu'une limite de secteur passe sous le bras, les changements d'état parvenus à échéance sont signalés au module " mémoire centrale ".

3.1.4. LE MODULE "SUPERVISEUR"

Ce module a pour fonctions

- d'assurer la synchronisation du système
- de tenir à jour un historique des programmes selon les mesures que l'on veut effectuer sur le comportement du simulateur et selon les algorithmes d'ordonnancement des programmes.
- d'attribuer l'unité centrale aux programmes.

L'historique tenu à jour pour chaque programme est composé de son temps d'exécution, de son temps d'attente pour transfert de pages et du nombre de ses défauts de pages cumulés pendant une durée déterminée. A partir de ces informations le taux de défauts de page, le temps de réponse pour chaque programme et l'efficacité du système sont calculés.

3.1.5. LE MODULE "ALGORITHME DE REMPLACEMENT DE PAGES"

Ce module, activé par le module "modèle de mémoire centrale", analyse les données de ce dernier ainsi que l'état des programmes contenu dans le module "comportement des programmes", l'état des files du module

"mémoire secondaire", et d'autres informations qu'il peut contenir lui-même. Eventuellement, l'état d'avancement de chaque programme peut être consulté dans les tableaux du module "superviseur".

Dans le cas de la multiprogrammation, ce module est également responsable de la partition de la mémoire entre les programmes activés. Cette partition est ajustée dynamiquement. Comme nous allons le voir en détail, lors d'un défaut de page, l'ARP prend en considération la totalité de la mémoire sans négliger l'individualité des programmes. Un mécanisme de régulation empêche qu'un des programmes ne soit trop brimé par les autres.

CAS DU SYSTEME MONOPROGRAMME.

Lorsqu'un seul programme est activé, l'ARP procède de la manière suivante :

- En fonction de l'état de la mémoire et de l'historique du programme est construit l'ensemble ordonné $\mathbb{T}_\ell$ de ℓ pages (où ℓ est un paramètre du système) susceptibles d'être déchargées. $\mathbb{T}_\ell$ est retenu sous forme d'une liste $\mathcal{L}_1$ ordonnée selon la priorité propre de l'algorithme utilisé (variante de FIFO, LRU ou RAND); par exemple, si l'ARP est FIFO(ℓ), le premier élément de $\mathcal{L}_1$ est le numéro de la page ayant le plus ancien temps de résidence ininterrompue en mémoire.

- Pour chacune des pages de $\mathcal{L}_1$ est calculée la date de fin des transferts au cas où cette page serait choisie. Pour la gestion du disque aucune page tampon n'est réservée en mémoire ce qui oblige à effectuer le déchargement avant le chargement de la page faisant défaut.

A la suite de ces calculs une liste $\mathcal{L}_2$ des pages dont le remplacement minimiserait la date d'échéance des transferts est extraite de $\mathcal{L}_1$. Son ordre est induit par l'ordre de $\mathcal{L}_1$.

- La page à remplacer est l'élément en tête de $\mathcal{L}_2$, cette liste pouvant d'ailleurs être réduite à un seul élément. Remarquons que la page remplacée est, parmi les pages de $\mathcal{T}_\ell$ minimisant le temps de réponse du disque, celle qui a la plus faible priorité.

CAS DU SYSTEME MULTIPROGRAMME

Supposons que d programmes (P_i) ($i = 1, \dots, d$) soient en compétition pour les m pages de la mémoire centrale. Au programme P_i ($i = 1, \dots, d$) est associée par l'ARP une liste $\mathcal{A}_i$ contenant par ordre de priorité (LRU, FIFO...) ses pages présentes en mémoire centrale. A des fins de régulation, les n_i dernières pages de $\mathcal{A}_i$ ne pourront être remplacées (les n_i sont des coefficients attribués par le superviseur et tels que $n_1 + n_2 + \dots + n_d < m$). A chaque défaut de page l'ARP procède de la manière suivante :

- Les éléments de $\mathcal{A}_i$ autres que les n_i derniers constituent une sous liste $\mathcal{L}_1^i$ des pages du programme P_i susceptibles d'être remplacées.

- Soit v l'indice du programme dont une page a été remplacée lors du précédent défaut de page. La liste $\mathcal{L}_1$ est obtenue par concaténation des sous listes $\mathcal{L}_1^i$ ($i = 1, \dots, d$), la sous liste placée en tête étant $\mathcal{L}_1^{v+1}$.

$$\mathcal{L}_1 = \mathcal{L}_1^{v+1} . \mathcal{L}_1^{v+2} . \dots . \mathcal{L}_1^d . \mathcal{L}_1^1 . \dots . \mathcal{L}_1^v$$

- La page à remplacer est obtenue à partir de $\mathcal{L}_1$ de la même manière que dans le cas du système monoprogrammé.

Notons que le respect d'un seuil minimum n_1 dans l'allocation dynamique de mémoire centrale est une méthode fréquemment employée pour résoudre le problème - analogue - de la gestion d'un "pool" de tampons [6]

3.2. RESULTATS RELATIFS A DES SYSTEMES MONOPROGRAMMES

Comme nous l'avons déjà signalé, le modèle à liste LRU a été choisi pour représenter le comportement des programmes. Nous avons toutefois introduit une légère modification dans l'interprétation du processus $(R_t)_{t \in \mathbb{N}}$. A chaque modèle de comportement est associé un entier INST. Nous considérerons qu'entre les références R_t et R_{t+1} ont été exécutés $(INST - 1)$ accès à la page qui fut déjà l'objet de la référence R_t . Avec cette convention, R_t est la $(t \times INST)^{\text{ème}}$ référence qui se produit et elle précède de INST unités de temps la référence R_{t+1} . Cette variation sur la définition du modèle à liste LRU modifie seulement l'échelle des temps et son importance n'apparaît donc qu'en relation avec le temps d'accès à la mémoire secondaire.

Les programmes considérés ont tous 20 pages virtuelles ($n = 20$). Les valeurs numériques retenues pour la distribution (α_i) sont respectivement les suivantes :

Programme A

$$\begin{aligned}\alpha_i &= C_1 \times 9 && \text{si } 1 \leq i \leq 4 \\ \alpha_i &= C_1 \times 2 && \text{si } 5 \leq i \leq 8 \\ \alpha_i &= C_1 \times 1 && \text{si } 9 \leq i \leq 20\end{aligned}$$

(C_1 est la constante de normalisation)

et INST = 1500

Programme B

$$\begin{array}{llll}
 \alpha_i = C_2 \times 200 & \text{si} & 1 \leq i \leq 2 \\
 \alpha_i = C_2 \times 100 & \text{si} & 3 \leq i \leq 4 \\
 \alpha_i = C_2 \times 50 & \text{si} & 5 \leq i \leq 6 \\
 \alpha_i = C_2 \times 15 & \text{si} & 7 \leq i \leq 12 \\
 \alpha_i = C_2 \times 1 & \text{si} & 13 \leq i \leq 20
 \end{array}$$

et INST = 100.

Programme GEOM (γ) (avec $0 < \gamma < 1$)

La distribution (α_i) est géométrique tronquée de rapport γ :

$$\alpha_i = C_3 \gamma^i$$

$$\text{avec } C_3 = \frac{1 - \gamma}{1 - \gamma^{n+1}}$$

et INST = 1500.

Ces programmes s'exécutent tous dans une mémoire centrale de 13 pages. La mémoire secondaire est un disque à trois secteurs dont la période de révolution est de 30 000 unités de temps.

Pour toutes les grandeurs qui nous intéressent, nous avons cherché à évaluer des valeurs moyennes, cette exigence fixant la durée minimum de la simulation. La figure 13 montre le taux de défauts de page sur $[0, t]$, c'est-à-dire la fraction $F(t)$ des références exécutées dans cet intervalle qui ont provoqué un défaut de page. L'oscillation de cette fonction est inférieure à 3% dès que t devient supérieur à 60×10^6 . Cette durée s'est généralement révélée suffisante ; néanmoins certaines valeurs ont nécessité que l'on simule 100 ou 120 millions d'accès à la mémoire principale.

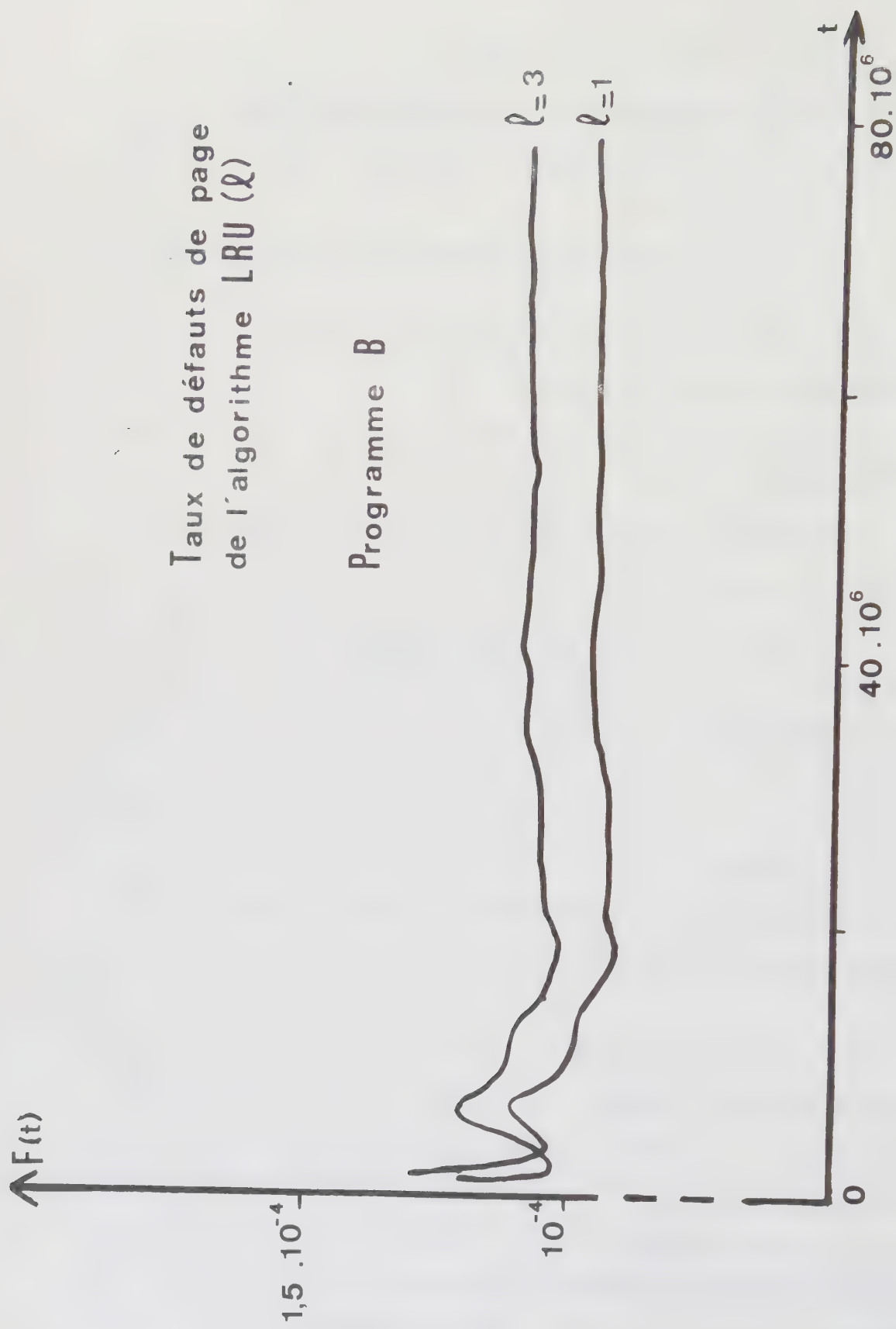
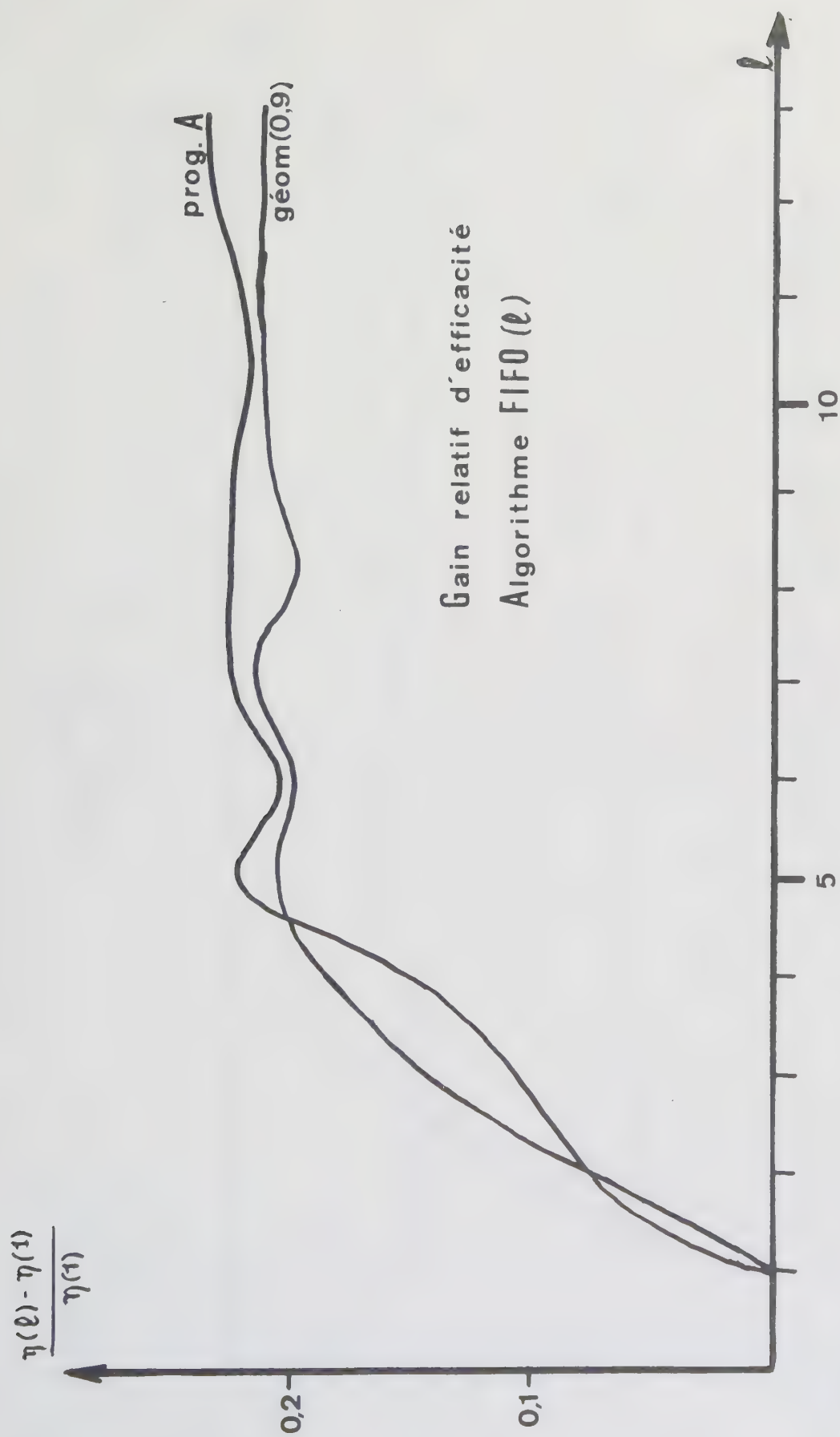


FIGURE 13



Gain relatif d'efficacité
 Algorithme FIFO (l)

FIGURE 14

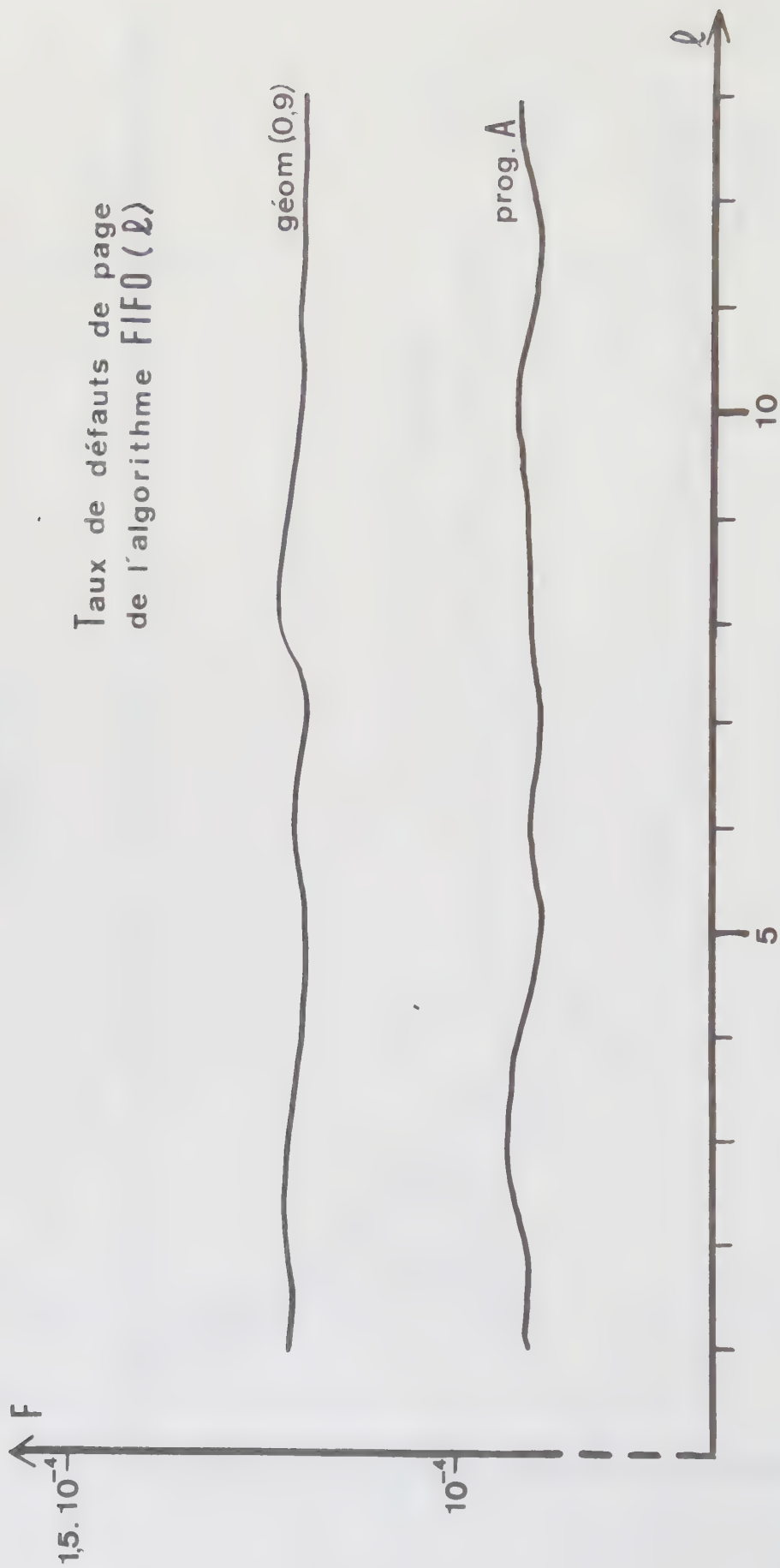


FIGURE 15

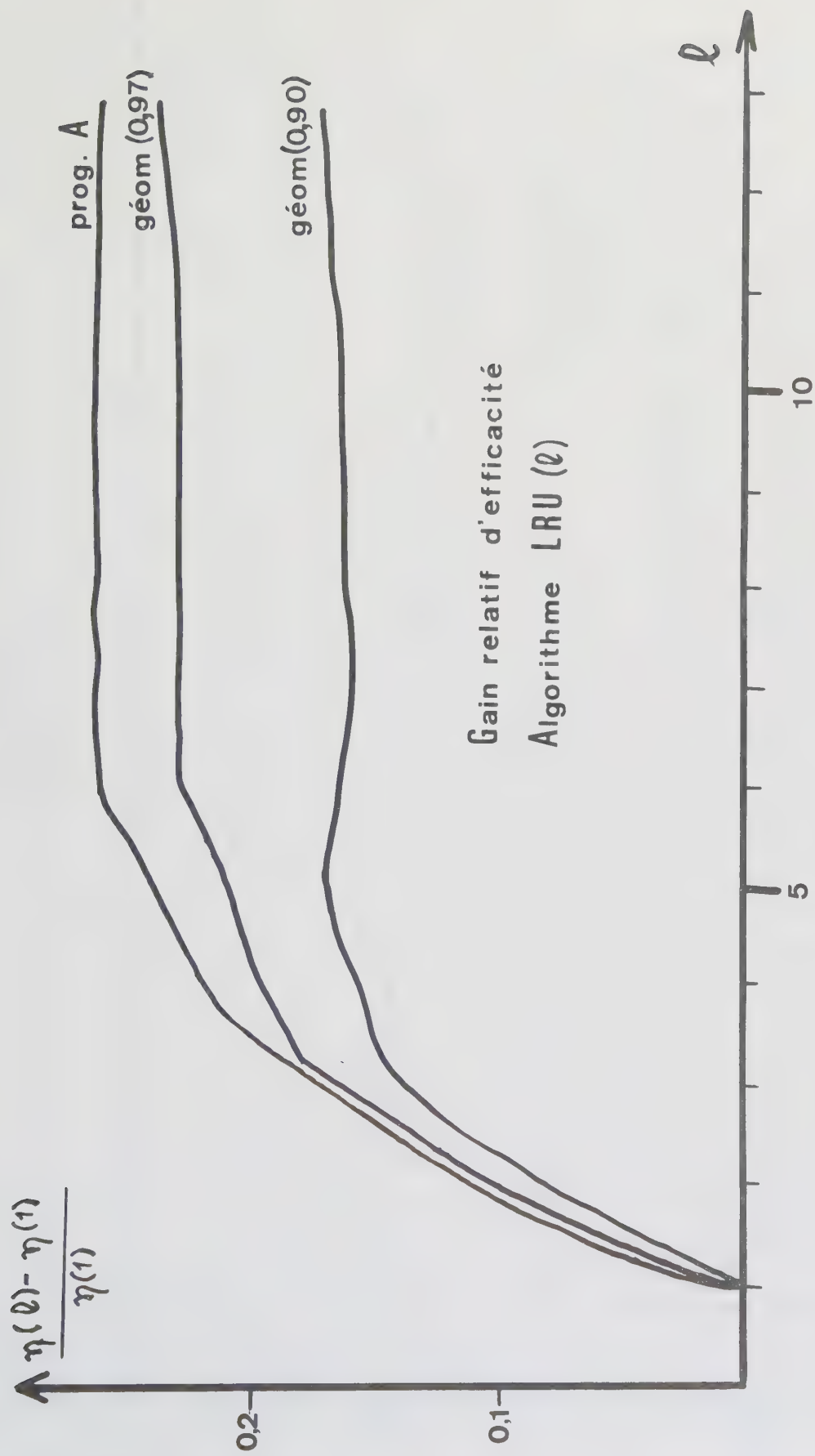


FIGURE 16

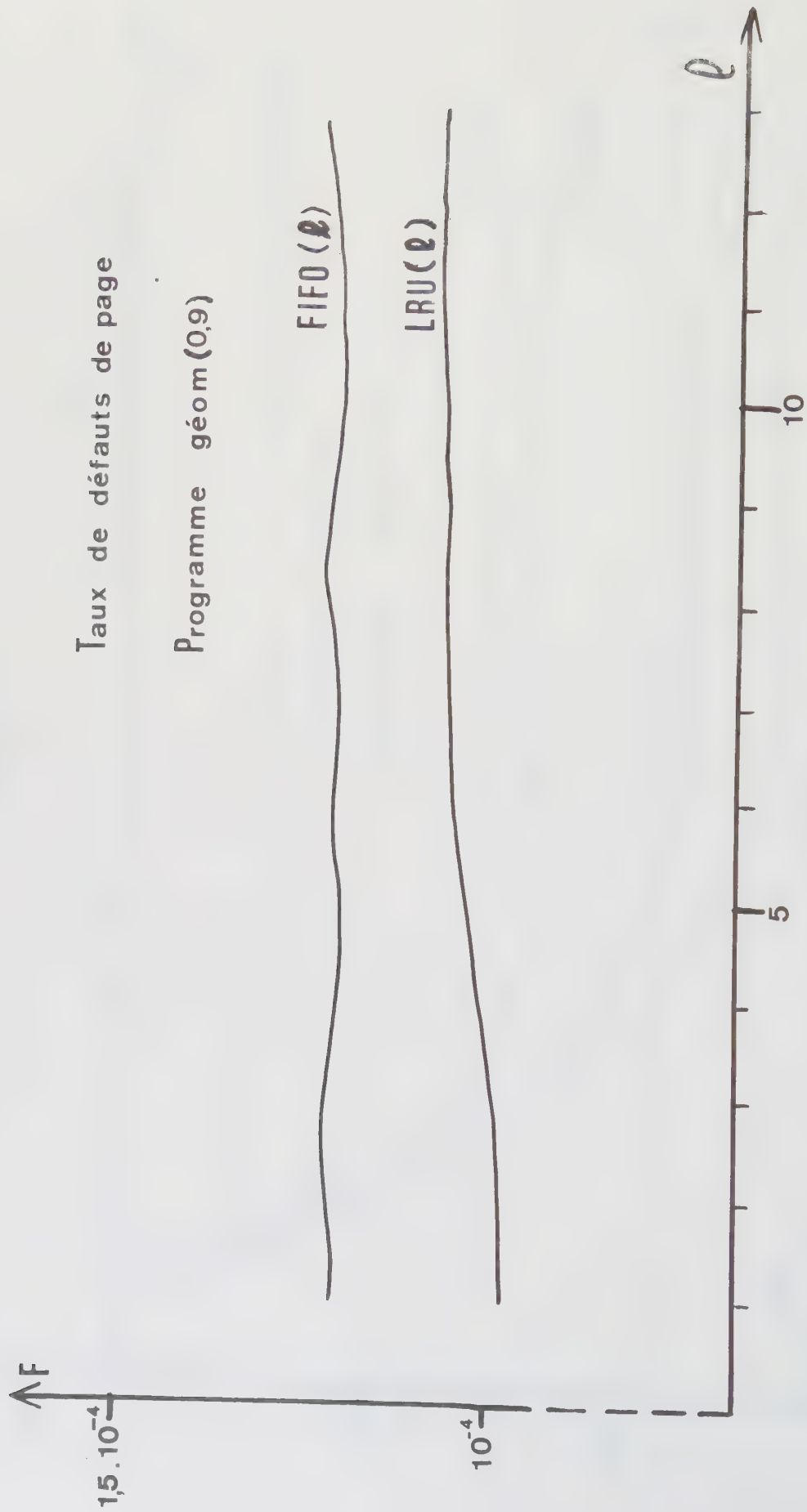


FIGURE 17

Comme au paragraphe 2.3., nous nous intéressons à l'efficacité du système, c'est-à-dire à la probabilité η que l'unité centrale soit active. La variation en fonction de ℓ de l'amélioration relative $\frac{\eta(\ell) - \eta(1)}{\eta(1)}$ est illustrée sur les figures 14 (algorithme FIFO(ℓ)) et 16 (algorithme LRU(ℓ)). Le gain apporté par la gestion globale de la mémoire centrale et du disque est très sensible, voire considérable. Remarquons que l'efficacité $\eta(\ell)$ est toujours supérieure à $\eta(1)$; en outre pour toutes les valeurs de ℓ supérieures à 5, $\eta(\ell)$ est proche de sa valeur optimale. Ce fait est important sur le plan pratique puisqu'il nous dispense d'estimer de manière précise la valeur optimale de ℓ ; le seul inconvénient d'une valeur trop élevée de ℓ réside dans l'accroissement du temps de mise en oeuvre de l'algorithme (overhead). Cette situation mérite un approfondissement, puisque dans le cas des algorithmes globaux à préchargement partiel, l'efficacité $\eta(\ell)$ décroît lorsque ℓ a dépassé sa valeur optimale et peut même devenir inférieure à $\eta(1)$ (cf figure 4). Pour ces algorithmes à préchargement partiel le choix d'une grande valeur de ℓ permet le déchargement de pages ayant une forte probabilité d'être référencées : la dégradation du taux de défauts de page qui en résulte ne peut être contrebalancée par le gain sur le temps de réponse du disque. Mais les algorithmes FIFO(ℓ) et LRU(ℓ) sont un peu plus élaborés. En raison du petit nombre de secteurs que comporte le disque, la probabilité est grande que la liste $\mathcal{L}_2$, décrite au paragraphe 3.1.4, ne soit pas réduite à un seul élément. Or la page à remplacer n'est pas choisie au hasard dans cette liste : elle en est le premier élément, c'est-à-dire l'élément le plus anciennement référencé (algorithme LRU(ℓ)) ou l'élément ayant le plus long temps de résidence ininterrompue en mémoire centrale (algorithme FIFO(ℓ)). Par conséquent, dans de nombreuses occasions l'algorithme LRU(ℓ) (resp. FIFO(ℓ)) prendra la même décision que LRU(ℓ')

(resp. $FIFO(\ell')$) avec $\ell' < \ell$. Cette explication est corroborée par l'allure des courbes représentant le taux de défauts de page : elles varient peu avec ℓ . A cet égard on pourra comparer les figures 15 et 17 d'une part, la figure 3 d'autre part.

Notons enfin que la situation illustrée par la figure 17 (taux de défauts de page de $FIFO(\ell)$ supérieur à celui de $LRU(\ell)$) a été rencontrée avec chacun des programmes dont nous avons simulé l'exécution. Elle est due au choix du modèle de comportement.

3.3. RESULTATS RELATIFS A DES SYSTEMES MULTIPROGRAMMES

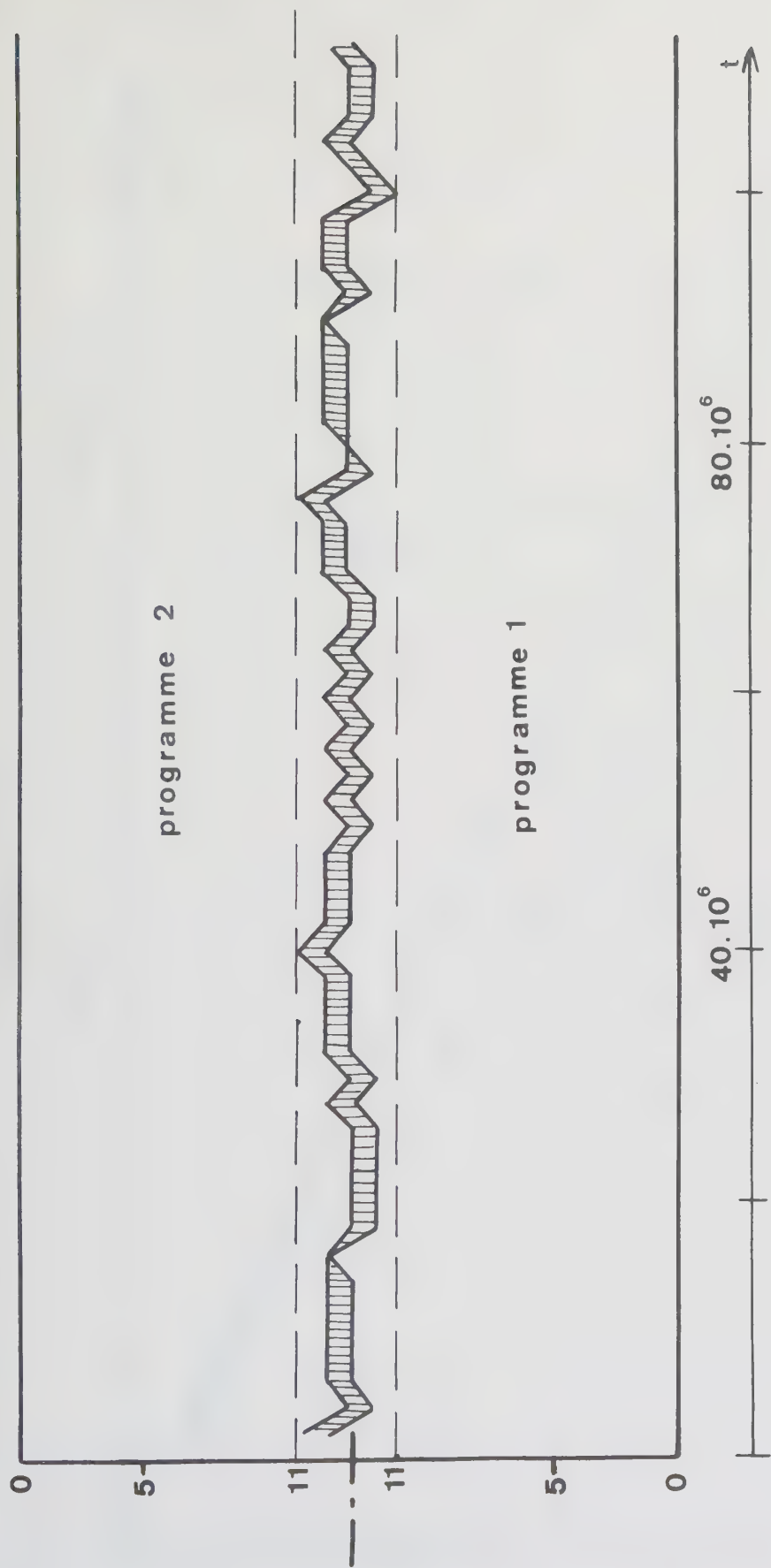
Nous considérons dans ce paragraphe d programmes ($d = 2$ ou 3) qui s'exécutent dans une même zone de mémoire. Chacun de ces programmes a un espace d'adresse de vingt pages et est conforme au modèle $LRU \text{ GEOM}(0, 9)$. La taille de la mémoire centrale est de $m = (13 \times d)$ pages physiques. L'algorithme d'attribution de cette ressource a été décrit en 3.1.5..

Il est nécessaire de définir le paramètre ℓ qui jusqu'ici n'a été considéré qu'en monoprogrammation ; très naturellement nous poserons

$$\ell = \frac{m - \sum_{i=1}^d n_i}{d} \quad (n_i \text{ est défini au paragraphe 3.1.5})$$

Après traitement de défauts de page, les programmes sont réactivés selon la règle Premier Prêt - Premier Activé.

Au cours des simulations, nous avons relevé, toutes les 2×10^6 unités de temps, la partition de la mémoire. La figure 18 représente l'un des historiques ainsi obtenus ($d = 2$; $\ell = 2$; les hachures indiquent des pages physiques dont le contenu était en train d'être modifié par un



Evolution dans le temps du partage de la mémoire

FIGURE 18

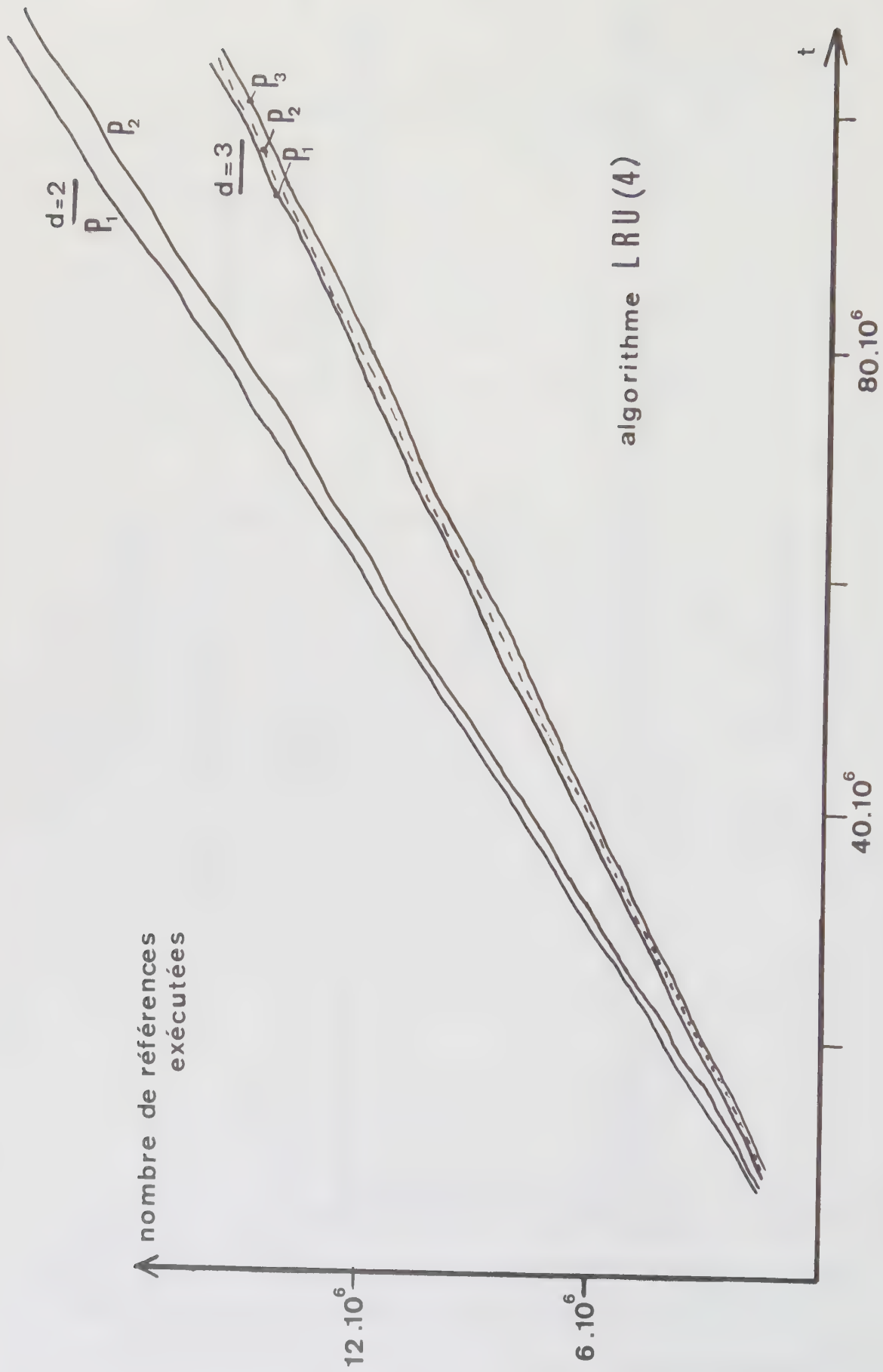


FIGURE 19

remplacement). On notera qu'il est peu fréquent qu'une des zones soit restreinte à sa taille minimum n_i .

La figure 19 montre le temps de calcul effectif dont a bénéficié chaque programme au cours d'un intervalle de temps de durée t . Que le niveau de multiprogrammation soit 2 ou 3, les courbes obtenues sont très voisines l'une de l'autre, ce qui montre que la politique de gestion n'introduit pas de discrimination entre des programmes ayant les mêmes caractéristiques statistiques.

ANNEXE

DEMONSTRATION DU LEMME

Récrivons E en ordonnant la sommation $\sum_{*}$ suivant la valeur des sommes σ ,

$$\sigma = n_{i+y_1} + \dots + n_{i+y_p}$$

$$E = \sum_{i=1}^S \frac{(n-\ell-1)!}{S^n n!} \sum_{*} n_{i+y_1} \frac{(n_{i+y_1} + n_{i+y_2} + \dots + n_{i+y_p} - 1)!}{(n_{i+y_1} + \dots + n_{i+y_p} - \ell - 1)!} \frac{n!}{n_1! n_2! \dots n_S!}$$

$$E = \sum_{i=1}^S \frac{(n-\ell-1)!}{S^n n!} \sum_{\sigma=\ell+1}^n \frac{(\sigma-1)!}{(\sigma-\ell-1)!} \sum_{**} n_{i+y_1} \frac{n!}{n_1! n_2! \dots n_S!}$$

la sommation $\sum_{**}$ étant étendue à tous les S-uples d'entiers non-négatifs $(n_1, n_2, \dots, n_S)$ tels que

$$\sum_{i=1}^S n_i = n,$$

les entiers non-négatifs $n_{i+y_1}, \dots, n_{i+y_p}$ vérifiant en outre :

$$\sum_{j=1}^p n_{i+y_j} = \sigma$$

En utilisant la partition de l'ensemble $\{1, \dots, S\}$ en ses deux

sous-ensembles :

$$\{y_1, y_2, \dots, y_p\}$$

et

$$\{y_{p+1}, \dots, y_S\}$$

nous pouvons remplacer les deux conditions ci-dessus par les suivantes :

$$n_{i+y_1} + n_{i+y_2} + \dots + n_{i+y_p} = \sigma$$

$$n_{i+y_{p+1}} + n_{i+y_{p+2}} + \dots + n_{i+y_S} = n - \sigma$$

Soit $E_1(\sigma)$, $E_2(\sigma)$, l'ensemble des solutions entières non-négatives des deux dernières équations. En remarquant qu'on peut toujours écrire :

$$n_1! n_2! \dots n_S! = (n_{i+y_1}! \dots n_{i+y_p}!)(n_{i+y_{p+1}}! \dots n_{i+y_S}!)$$

nous transformons la sommation $\sum$ en :

$$\begin{aligned} \sum_{**} n! \frac{n_{i+y_1}}{n_1! n_2! \dots n_S!} &= \sum_{**} n! \frac{n_{i+y_1}}{n_{i+y_1}! \dots n_{i+y_p}!} \times \frac{1}{n_{i+y_{p+1}}! \dots n_{i+y_S}!} \\ &= n! \sum_{E_1(\sigma)} \frac{n_{i+y_1}}{n_{i+y_1}! \dots n_{i+y_p}!} \sum_{E_2(\sigma)} \frac{1}{n_{i+y_{p+1}}! \dots n_{i+y_S}!} \end{aligned}$$

Rappelons le théorème suivant :

Théorème

La somme $\sum \frac{1}{k_1! k_2! \dots k_q!}$, étendue à tous les q-uples $(k_1, k_2, \dots, k_q)$

d'entiers non-négatifs vérifiant :

$$k_1 + k_2 + \dots + k_q = c$$

est égale à : $\frac{q^c}{c!}$

De ce résultat, connu sous le nom de formule du multinôme, nous déduisons la valeur de chacune des sommes $\sum_{E_1(\sigma)}$ et $\sum_{E_2(\sigma)}$:

$$\sum_{E_2(\sigma)} \frac{1}{n_{i+y_p}! \dots n_{i+y_s}!} = \frac{(S-p)^{n-\sigma}}{(n-\sigma)!}$$

et

$$\sum_{E_1(\sigma)} \frac{n_{i+y_1}}{n_{i+y_1}! \dots n_{i+y_p}!} = \sum_{E_1(\sigma)} \frac{1}{(n_{i+y_1}-1)! \dots n_{i+y_p}!} = \frac{p^{\sigma-1}}{(\sigma-1)!}$$

Revenant à E, nous obtenons :

$$E = \sum_{i=1}^S \frac{(n-l-1)!}{S^n n!} \sum_{\sigma=l+1}^n \frac{(\sigma-1)!}{(\sigma-l-1)!} \frac{n!(S-p)^{n-\sigma} p^{\sigma-1}}{(n-\sigma)! (\sigma-1)!}$$

$$E = \sum_{i=1}^S \frac{(n-l-1)!}{S^n} \sum_{\sigma=l+1}^n \frac{1}{(\sigma-l-1)!(n-\sigma)!} (S-p)^{n-\sigma} p^{\sigma-1}$$

Regroupons les termes de façon à faire apparaître le terme général du développement selon la formule du binôme de :

$$((S-p)+p)^{n-l-1} = \sum_{q=0}^{n-l-1} \binom{n-l-1}{q} (S-p)^{n-l-1-q} p^q = \sum_{q=l+1}^n \binom{n-l-1}{q-l-1} (S-p)^{n-q} p^{q-l-1}$$

$$E = \sum_{i=1}^S \frac{p^l}{S^n} \sum_{\sigma=l+1}^n \frac{(n-l-1)!}{(\sigma-l-1)!(n-\sigma)!} = (S-p)^{n-\sigma} p^{l-\sigma-1} = \sum_{i=1}^S \frac{p^l}{S^n} ((S-p)+p)^{n-l-1}$$

Finalement :

$$E = \sum_{i=1}^S \frac{p^l}{S^{l+1}} = \frac{p^l}{S^l}$$

ce qui est le résultat annoncé.

REFERENCES

- [1] WEINGARTEN, A. : The Eschenbach drum scheme -Comm. ACM 9, 7, 509-512
(1966)
- [2] COFFMAN, E.G. Jr : Analysis of a drum input/output queue under scheduled operation in a paged computer system -Journal ACM 16, 1, 73-90 (1971)
- [3] GELENBE, E., LENFANT, J., POTIER, D. : Analyse d'un algorithme de gestion simultanée Mémoire centrale-Disque de pagination - Acta Informatica,3,4,321-346 (1974)
- [4] VANTILBORGH, H : On random partially preloaded page replacement algorithms - MBLE, rapport R 202, Bruxelles (1972)
- [5] BURGEVIN, P., RICHY, H., TOUILLET, D. : Simulation d'une mémoire virtuelle paginée - Rapport de projet, I.N.S.A., Rennes (1973)
- [6] DIJKSTRA, E.W. : Information streams sharing a finite buffer-Information Processing Letters 1, 179-180 (1972)

CHAPITRE QUATRE

TAILLE DE L'ENSEMBLE DE TRAVAIL

1 - INTRODUCTION

1.1. LE CONCEPT D'ENSEMBLE DE TRAVAIL.

Pour formaliser la notion de "localité", P. Denning [1] a introduit le concept d'ensemble de travail (en anglais : Working set). Pour une exécution donnée d'un programme, l'ensemble de travail (E.T.) de fenêtre T est, à l'instant t , l'ensemble noté $W(t, T)$ des adresses qui ont été référencées dans l'intervalle de temps $]t-T, t]$. Sur un modèle de programme assez général, Denning et Schwartz [2] ont calculé l'espérance mathématique de la taille de cet ensemble de travail en fonction de la distribution de l'intervalle entre deux références successives à une même adresse. Toutefois la seule connaissance de cette espérance mathématique est insuffisante pour qui veut analyser des politiques d'allocation de ressources : dans ce cas l'efficacité du système dépend autant de la dispersion autour de la valeur moyenne que de cette valeur moyenne elle-même. Il est donc nécessaire de connaître la distribution de la taille $w(t, T)$ de l'E.T. ou, au moins, ses deux premiers moments. C'est ainsi que pour comparer l'efficacité d'une partition dynamique et d'une partition statique de la mémoire principale, Coffman et Ryan [3] ont admis que la distribution de la variable aléatoire $w(t, T)$ est presque normale. Cette approximation, bien que suggérée par plusieurs auteurs, n'a jamais été justifiée de manière convaincante.

L'objet de ce chapitre est le calcul de la distribution de taille de

l'E.T. sur deux modèles de comportement de programme : le modèle à références indépendantes et le modèle à liste LRU.

L'intérêt de la notion d'E.T. ne se limite pas à l'étude statistique du comportement local des programmes. Il est possible de l'utiliser pour définir des règles d'attribution de la mémoire centrale. Une politique de gestion de cette ressource est dite "par ensemble de travail" si l'ensemble des pages virtuelles d'un programme présentes en mémoire à l'instant t est son E.T. $W(t, T)$. Le choix d'une telle politique a les conséquences suivantes :

- . le nombre de pages physiques allouées à un programme est variable
- . un programme ne peut être activé que s'il existe en mémoire centrale une place disponible suffisante pour contenir son E.T.
- . si toutes les pages physiques sont occupées par une page virtuelle appartenant à un E.T. et qu'un défaut de page se produit, un programme doit être désactivé.

La régulation du système repose alors sur le choix du paramètre T - ou d'une famille de paramètres (T_i) si l'on a choisi de diversifier les prérogatives des programmes activés ou en attente. On ne dispose pas encore de résultats expérimentaux sur l'efficacité des politiques d'allocation par E.T. car aucun système doté des mécanismes adéquats n'est actuellement en service. Toutefois les études théoriques et les mises en oeuvre de mécanismes voisins (par exemple l'allocateur d'Unité centrale d'ESOPÉ) laissent attendre de bonnes performances. Un calculateur expérimental, construit à Los Alamos, a été doté pour un coût modique des dispositifs matériels permettant une gestion par E.T.

1.2. DEFINITIONS ET PROPRIETES GENERALES

Comme dans les chapîtres précédents, nous considérons un programme dont l'espace d'adresse est divisé en pages : $N = \{1, 2, \dots, n\}$. La suite des adresses référencées au cours d'une exécution est représentée par un processus stochastique $(R_t)_{t \in \mathbb{N}}$.

Définition 1

Soient t et T deux entiers positifs tels que $T \leq t$. Pour une réalisation

$$r_0 \ r_1 \ r_2 \dots r_t \dots$$

du processus (R_t) l'ensemble de travail de fenêtre T est, à l'instant t

$$W(t, T) = \{i \in \mathbb{N} \mid (\exists u \in \mathbb{N}) (t-T+1 \leq u \leq t) : r_u = i\}$$

La taille de $W(t, T)$ (notée $w(t, T)$) est son nombre d'éléments.

Définition 2

Le taux d'absence de page $m(t, T)$ est la probabilité que la page référencée à l'instant $t+1$ n'appartienne pas à l'E.T. $W(t, T)$:

$$\begin{aligned} m(t, T) &= \Pr [r_{t+1} \notin W(t, T)] \\ &= \Pr [w(t+1, T+1) - w(t, T) = 1] \end{aligned}$$

Rappelons le résultat établi par Denning et Schwartz [2] sur la taille de l'E.T. des programmes conformes au modèle du chapitre 1, paragraphe 2.1. . Soit $f(x)$ la probabilité qu'un intervalle entre deux références consécutives à une même page soit de longueur x . Avec les notations introduites au paragraphe cité ci-dessus, nous pouvons écrire :

$$(1) \quad f(x) = \sum_{i=1}^n \frac{1}{X_i} f_i(x)$$

Désignons par F la fonction cumulative associée à f :

$$(2) \quad F(y) = \sum_{x=1}^y f(x)$$

Théorème 1

Soit un programme dont le comportement obéit aux hypothèses DS1 à DS3. Pour tous entiers positifs t et T tels que $T \leq t$, les deux égalités ci-dessous sont vérifiées :

$$(3) \quad \overline{w(t, T)} = \sum_{y=0}^{T-1} (1-F(y))$$

$$(4) \quad m(t, T) = 1 - F(T)$$

On remarque que la taille moyenne de l'E.T. et le taux d'absence de page ne dépendent pas de t . En outre

$$m(T) = \overline{w}(T+1) - \overline{w}(T)$$

2 - DISTRIBUTION DE LA TAILLE D'UN ENSEMBLE DE TRAVAIL

2.1. PROGRAMMES CONFORMES AU MODELE A LISTE LRU

Considérons un programme se comportant selon le modèle à liste LRU, à distances indépendantes ; nous désignerons par α_i la probabilité que la i^e page de la liste LRU soit la prochaine à être référencée. L'E.T.

$W(t, T)$ a exactement k éléments si et seulement si la chaîne

$$D_{t-T+1} \quad D_{t-T+2} \quad \dots \quad D_t$$

des distances associées aux références exécutées entre $t-T+1$ et t est de la forme

$$(5) \quad \delta_1 (1)^{r_1} \quad \delta_2 (1,2)^{r_2} \quad \delta_3 (1,2,3)^{r_3} \quad \dots \quad \delta_k (1,2,\dots,k)^{r_k}$$

où

- . δ_i est une distance supérieure ou égale à i
- . les r_i sont des entiers positifs ou nuls dont la somme est $T-k$
- . $(1, 2, \dots, \ell)^r$ désigne une suite de r distances toutes non supérieures à ℓ

Par suite de l'indépendance des variables aléatoires D_t , la probabilité que l'E.T. soit de taille k ($1 \leq k \leq \min(n, T)$) est donc :

$$(6) \quad \Pr [w(t, T) = k] = \Pr [D_{t-T+1} \geq 1] \times \Pr [D_{t-T+r_1+2} \geq 2] \\ \times \dots \times \Pr [D_{t-T+r_k} \geq k] \times P(\alpha_1, \alpha_2, \dots, \alpha_k)$$

où l'on désigne par $P(\alpha_1, \alpha_2, \dots, \alpha_k)$ la somme des monômes

$$\alpha_1^{r_1} (\alpha_1 + \alpha_2)^{r_2} \dots (\alpha_1 + \alpha_2 + \dots + \alpha_k)^{r_k} \text{ pour tous les } k\text{-uples } (r_i)$$

d'entiers positifs ou nuls tels que $\sum_{i=1}^k r_i = T - k$.

$$\text{Posons } S_\ell = \sum_{i=1}^{\ell} \alpha_i \quad (0 \leq \ell \leq n). \text{ Puisque } \Pr [D \geq \ell] = 1 - S_{\ell-1},$$

l'égalité (6) équivaut à

$$(7) \quad \Pr [w(t, T) = k] = (1 - S_1)(1 - S_2) \dots (1 - S_{k-1}) P(\alpha_1, \alpha_2, \dots, \alpha_k)$$

Comme le développement en série entière de $\frac{1}{1 - S_\ell z}$ est

$$1 + (\alpha_1 + \alpha_2 + \dots + \alpha_\ell)z + (\alpha_1 + \alpha_2 + \dots + \alpha_\ell)^2 z^2 + (\alpha_1 + \alpha_2 + \dots + \alpha_\ell)^3 z^3 + \dots$$

le nombre $P(\alpha_1, \alpha_2, \dots, \alpha_k)$ est le coefficient de z^{T-k} dans le développement en série entière de la fonction rationnelle :

$$Q(z) = \frac{1}{(1-S_1z) (1-S_2z) \dots (1-S_kz)}$$

Tous les pôles de Q étant d'ordre 1 d'après l'hypothèse LRU3, la décomposition en éléments simples de cette fonction est

$$(8) \quad Q(z) = \sum_{i=1}^k \frac{S_i^{k-1}}{\prod_{\substack{1 \leq j \leq k \\ j \neq i}} (S_i - S_j)} \frac{1}{1 - S_i z}$$

En utilisant cette expression de Q pour calculer les coefficients de son développement en série entière nous obtenons :

$$(9) \quad P(\alpha_1, \alpha_2, \dots, \alpha_k) = \sum_{i=1}^k \frac{S_i^{T-1}}{\prod_{\substack{1 \leq j \leq k \\ j \neq i}} (S_i - S_j)}$$

Enonçons ce résultat :

Théorème 2

La distribution de la taille de l'E.T. d'un programme se comportant suivant le modèle LRU est déterminée pour tout entier k positif et au plus égal à min(n, T) par

$$\text{Pr } [w(t, T) = k] = (1-S_1) (1-S_2) \dots (1-S_{k-1}) \sum_{i=1}^k \frac{S_i^{T-1}}{\prod_{\substack{1 \leq j \leq k \\ j \neq i}} (S_i - S_j)}$$

Cette formule permet de faire rapidement le calcul effectif de la distribution de w(t, T). Remarquons en particulier que

$$m(t, T) = \sum_{k=1}^{\min(n, t)} (1-S_k) \text{Pr } [w(t, T) = k]$$

2.2. PROGRAMMES CONFORMES AU MODELE A REFERENCES INDEPENDANTES

Considérons un programme de n pages virtuelles représenté par un modèle à références indépendantes ; conformément aux notations adoptées dans les chapitres précédents, nous désignerons par β_i ($1 \leq i \leq n$) la probabilité qu'une référence concerne la page i .

Pour un programme ayant un comportement à références indépendantes, la probabilité que la chaîne des pages référencées entre $t-T+1$ et t soit

$$\gamma = i_1 i_2 i_3 \dots i_T$$

est égale à

$$\begin{aligned} \pi(\gamma) &= \Pr [R_{t-T+1} = i_1] \times \Pr [R_{t-T+2} = i_2] \times \dots \times \Pr [R_t = i_T] \\ (10) \quad &= \beta_{i_1} \times \beta_{i_2} \times \dots \times \beta_{i_T} \end{aligned}$$

Par suite la probabilité que l'E.T. $W(t, T)$ ait pour taille k ($1 \leq k \leq \min(n, T)$) est

$$(11) \quad \Pr [w(t, T) = k] = \sum_{*} \pi(\gamma),$$

la sommation $\sum_{*}$ étant étendue à toutes les chaînes de T éléments de N comportant exactement k éléments distincts. Pour évaluer (11) nous allons utiliser une méthode proche de celle qui est employée pour démontrer le théorème d'inversion de Möbius [6].

Nous dirons qu'une chaîne γ de N^T a la propriété $A(i)$ si elle ne contient pas la page i ; $i_1, i_2, \dots, i_q$ étant q pages distinctes, nous désignerons par $N_{i_1 i_2 \dots i_q}$ l'ensemble des chaînes de N^T ayant à la fois les

propriétés $A(i_1), A(i_2) \dots A(i_q)$ (et éventuellement d'autres propriétés) et par $\pi_{i_1 i_2 \dots i_q}$ la probabilité de cet ensemble. Il est clair que :

$$(12) \quad \pi_{i_1 i_2 \dots i_q} = \Pr [N_{i_1 i_2 \dots i_q}] = \sum_{\gamma \in N_{i_1 i_2 \dots i_q}} \pi(\gamma) \\ = (1 - \beta_{i_1} - \beta_{i_2} - \dots - \beta_{i_q})^T$$

Nous allons montrer que pour tout $1 \leq k \leq \min(n, T)$:

$$(13) \quad \Pr [w(t, T) = k] = \sum_{r=n-k}^n (-1)^{r-n+k} \binom{r}{n-k} \sum_{i_1 < i_2 < \dots < i_r} \pi_{i_1 i_2 \dots i_r}$$

(La sommation $\sum_{i_1 < i_2 < \dots < i_r}$ est une sommation sur toutes les combinaisons de r entiers choisis dans $\{1, 2, \dots, n\}$)

En effet, soit γ une chaîne de T références. Trois éventualités sont à envisager :

a) si γ contient strictement plus de k termes différents, elle a moins de $(n-k)$ propriétés du type $A(i)$: elle ne figure donc dans aucun des ensembles $N_{i_1 i_2 \dots i_q}$ dont les probabilités sont prises en compte dans le membre de droite de l'égalité (13).

b) si γ contient exactement k termes différents, c'est-à-dire si elle possède les propriétés $A(j_1), A(j_2), \dots, A(j_{n-k})$ à l'exclusion de toute autre, sa probabilité $\pi(\gamma)$ contribue à $\pi_{j_1 j_2 \dots j_{n-k}}$ (qui est affecté du coefficient $\binom{n-k}{n-k} = 1$) et seulement à ce terme

c) si γ a exactement s propriétés du type $A(i)$ ($s > n - k$)

sa contribution à

$$(-1)^{r-n+k} \binom{r}{n-k} \sum_{i_1 < i_2 \dots < i_r} \pi_{i_1 i_2 \dots i_r} \quad (n-k \leq r \leq s)$$

est égale à $(-1)^{r-n+k} \binom{r}{n-k} \binom{s}{r} \pi(\gamma)$. La contribution totale d'une telle chaîne au membre de droite de l'égalité (13) est donc :

$$(14) \quad (-1)^{k-n} \pi(\gamma) \sum_{r=n-k}^s (-1)^r \binom{s}{r} \binom{r}{n-k}$$

En dérivant $(n-k)$ fois l'identité binômiale

$$(1+x)^s = \sum_{r=0}^s \binom{s}{r} x^r$$

en divisant par $(n-k)!$ l'identité ainsi obtenue et en posant $x=-1$, nous obtenons :

$$\sum_{r=n-k}^s (-1)^r \binom{s}{r} \binom{r}{n-k} = \frac{1}{(n-k)!} \frac{d^{(n-k)}}{dx^{(n-k)}} (1+x)^s \Big|_{x=-1}$$

quantité qui est nulle puisque $s \neq n-k$. La contribution de γ au membre de droite de (13) est donc nulle également dans ce cas.

Il résulte de cette étude que le membre de droite de (13) est la somme des probabilités des chaînes de N^T qui contiennent exactement k éléments distincts : cette somme est égale à $\Pr [w(t, T) = k]$ d'après l'égalité (11). D'où le théorème :

Théorème 3

La distribution de la taille de l'E.T. d'un programme se comportant suivant le modèle à références indépendantes est déterminée pour tout entier k positif et au plus égal à $\min(n, T)$ par :

$$(15) \quad \Pr [w(t, T) = k] = \sum_{r=n-k}^n (-1)^{r-n+k} \binom{n-k}{r} U_r$$

où l'on a posé

$$U_r = \sum_{i_1 < i_2 \dots < i_r} (1 - \beta_{i_1} - \beta_{i_2} - \dots - \beta_{i_r})^T$$

et $U_0 = 1$

Corollaire

Pour un programme ayant un comportement à références indépendantes, la moyenne et la variance de la taille d'un E.T. sont respectivement égales à

$$(16) \quad \bar{w}(T) = n - U_1$$

et

$$(17) \quad \sigma^2(T) = -U_1^2 + (2n-1) U_1 + 2U_2$$

Pour démontrer ces deux formules, utilisons l'égalité (15) :

$$\Pr [w(T)=n] = \binom{0}{0} U_0 - \binom{1}{0} U_1 + \binom{2}{0} U_2 - \binom{3}{0} U_3 + \dots + (-1)^n \binom{n}{0} U_n$$

$$\Pr [w(T)=n-1] = \binom{1}{1} U_1 - \binom{2}{1} U_2 + \binom{3}{1} U_3 - \dots + (-1)^{n-1} \binom{n}{1} U_n$$

$$\Pr [w(T)=n-2] = \binom{2}{2} U_2 - \binom{3}{2} U_3 + \dots + (-1)^{n-2} \binom{n}{2} U_n$$

.....

$$\Pr [w(T)=n-\ell] = \binom{\ell}{\ell} U_{\ell} - \binom{\ell+1}{\ell} U_{\ell+1} + \dots + (-1)^{n-\ell} \binom{n}{\ell} U_n$$

.....

$$\Pr [w(T)=1] = \binom{n-1}{n-1} U_{n-1} - \binom{n}{n-1} U_n$$

$$\Pr [w(T)=0] = \binom{n}{n} U_n$$

En évaluant "par colonnes" la somme $\sum_{r=0}^n r \Pr [w(T)=r]$,

on obtient :

$$\begin{aligned} \overline{w}(T) &= \sum_{k=0}^n \sum_{\ell=0}^k (n-\ell) (-1)^{k+\ell} \binom{k}{\ell} U_k \\ (18) \quad &= \sum_{k=0}^n (-1)^k U_k \sum_{\ell=0}^k (n-\ell) (-1)^{\ell} \binom{k}{\ell} \end{aligned}$$

La sommation la plus à droite peut se scinder en deux :

$$\blacktriangleright n \sum_{\ell=0}^k (-1)^{\ell} \binom{k}{\ell} \text{ est égal à } n \text{ si } k = 0 \text{ sinon à } n(1-1)^k = 0$$

$$\blacktriangleright \sum_{\ell=0}^k (-1)^{\ell-1} \binom{k}{\ell} \text{ est la valeur pour } x = -1 \text{ de la dérivée de } (1+x)^k :$$

cette somme est donc nulle sauf si $k = 1$, auquel cas elle vaut 1.

Finalement l'égalité (18) implique

$$\overline{w}(T) = nU_0 - U_1 = n - U_1$$

ce qui est le résultat annoncé. Un calcul analogue nous permet de trouver le second moment de la distribution de $w(T)$.

En effet :

$$(19) \quad E(w^2(T)) = \sum_{k=0}^n (-1)^k U_k \sum_{\ell=0}^k (n^2 - 2\ell + \ell^2) (-1)^\ell \binom{k}{\ell}$$

Compte tenu de ce que $\sum_{\ell=0}^k \ell^2 (-1)^\ell \binom{k}{\ell}$ est la valeur au point $x = -1$ de

$-\frac{d}{dx} \left[x \frac{d}{dx} (1+x)^k \right]$, l'égalité (19) conduit à

$$E(w^2(T)) = n^2 U_0 - U_1 + 2U_2$$

La variance de la taille de l'E.T. est donc

$$\sigma^2(T) = E(w^2(T)) - E(w(T))^2 = -U_1^2 + (2n - 1) U_1 + 2U_2$$

Remarque sur une autre approche possible

Il est possible de calculer cette distribution de $w(t, T)$ par une méthode analogue à celle qui a été employée pour les programmes suivants un modèle de comportement à liste LRU. Calculons la probabilité que $w(t, T)$ soit égal à k ($1 \leq k \leq \min(n, T)$). L'ensemble de travail $W(t, T)$ est égal à $\{1, 2, \dots, k\}$ si et seulement si la chaîne des T références exécutées entre l'instant $t-T+1$ et l'instant t est

- soit une chaîne de la forme

$$(20) \quad 1^{r_1} 2^{r_2} 3^{r_3} \dots k^{r_k}$$

où

- . les r_i sont des entiers positifs ou nuls, de somme $T - k$
- . $(1, 2, \dots, \ell)^r$ désigne une suite de r références à des pages du sous ensemble $\{1, 2, \dots, \ell\}$

- soit une chaîne déduite d'une chaîne du type précédent par

une permutation sur $\{1, 2, \dots, k\}$

La probabilité qu'une suite de T références générées par le programme soit de la forme décrite en (20) est égale à

$$\beta_1 \beta_2 \dots \beta_k P(\beta_1, \beta_2, \dots, \beta_k)$$

où P désigne la même fonction qu'au paragraphe 2.1. Par additivité, il en résulte :

$$\Pr [w(t, T) = \{1, 2, \dots, k\}] = \beta_1 \beta_2 \dots \beta_k \sum_{*} P(\beta_{\sigma(1)}, \beta_{\sigma(2)}, \dots, \beta_{\sigma(k)})$$

la sommation $\sum_{*}$ étant étendue à toutes les permutations σ de $\{1, 2, \dots, k\}$.

De même

$$\Pr [w(t, T) = k] = \sum_{**} \beta_{i(1)} \beta_{i(2)} \dots \beta_{i(k)} P(\beta_{i(1)}, \beta_{i(2)}, \dots, \beta_{i(k)})$$

la sommation $\sum_{**}$ étant étendue à l'ensemble I_k des arrangements i des n premiers entiers positifs choisis k à k , c'est-à-dire à l'ensemble des injections i de $\{1, 2, \dots, k\}$ dans $\{1, 2, \dots, n\}$.

Du calcul de P qui a été effectué au paragraphe 2.1., on tire donc l'expression suivante pour la distribution de $w(t, T)$:

$$(21) \quad \Pr [w(t, T) = k] = \sum_{i \in I_k} \beta_{i(1)} \beta_{i(2)} \dots \beta_{i(k)} \sum_{r=1}^k \frac{S_{r,i}^{T-1}}{\prod_{\substack{1 \leq s \leq k \\ s \neq r}} (S_{r,i} - S_{s,i})}$$

où l'on a posé : $S_{\ell,i} = \beta_{i(1)} + \beta_{i(2)} + \dots + \beta_{i(\ell)}$.

En raison du nombre élevé d'arrangements ($|I_k| = \frac{n!}{(n-k)!}$)

la formule (15) est préférable pour le calcul effectif de la distribution de $w(t, T)$.

2.3. RESULTATS NUMERIQUES

Sur les figures 1 et 2 est rapportée la taille de l'E.T. de deux comportements de programme : le premier de ces comportements est à références indépendantes, le second à liste LRU. La valeur de $\text{Pr} [w(T) = k]$

$$(1 \leq k \leq n)$$

y est comparée à son approximation gaussienne :

$$\Pi_k = \Phi\left(\frac{k+1/2 - \bar{w}(T)}{\sigma}\right) - \Phi\left(\frac{k-1/2 - \bar{w}(T)}{\sigma}\right)$$

où l'on désigne par σ l'écart-type de $w(T)$ et par Φ la fonction de répartition de la distribution normale réduite :

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{t^2}{2}} dt$$

Sur ces figures, ainsi que sur les quelques centaines d'autres résultats de ce type que nous avons obtenus, l'accord entre $\text{Pr} [w(T) = k]$ et Π_k est très bon ; il est d'autant meilleur que n est plus grand. Toutefois lorsque T est très supérieur à n (par exemple : $T = 480$ sur la figure 2) la différence entre $\text{Pr} [w(T) = k]$ et Π_k peut être sensible : ceci n'est pas surprenant car on approxime alors par une distribution continue une distribution discrète dont seules quelques valeurs ont une probabilité non négligeable.

Les figures 3 et 4 représentent la variation en fonction de T de la moyenne et de l'écart-type de la taille de l'E.T. du programme déjà considéré sur la figure 2.

CARACTERISTIQUES DU PROGRAMME

Nombres de pages : $n = 15$

Modèles à références indépendantes

$\beta_i = C (16-i)$

(C est la constante de normalisation)

T = 15

k	1	2	3	4	5	6	7	8
Pr $[w(T)=k]$	0,000	0,000	0,000	0,000	0,002	0,024	0,112	0,260
Approximation gaussienne	0,000	0,000	0,000	0,000	0,003	0,027	0,114	0,257

k	9	10	11	12	13	14	15
Pr $[w(T)=k]$	0,316	0,204	0,068	0,011	0,001	0,000	0,000
Approximation gaussienne	0,311	0,202	0,071	0,013	0,001	0,000	0,000

T = 125

k	1	2	3	4	5	6	7	8
Pr $[w(T)=k]$	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
Approximation gaussienne	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

k	9	10	11	12	13	14	15	16
Pr $[w(T)=k]$	0,000	0,000	0,000	0,003	0,064	0,402	0,532	
Approximation gaussienne	0,000	0,000	0,000	0,001	0,062	0,462	0,427	0,048

FIGURE 1

Caractéristiques du programme

Nombre de pages n=40

Modèle à liste LRU

$$\begin{aligned} \alpha_i &= C \times 20 & \text{si } 1 \leq i \leq 8 \\ \alpha_i &= C \times 10 & \text{si } 9 \leq i \leq 20 \\ \alpha_i &= C \times 7 & \text{si } 21 \leq i \leq 23 \\ \alpha_i &= C \times 6 & \text{si } 24 \leq i \leq 26 \\ \alpha_i &= C \times 5 & \text{si } 27 \leq i \leq 29 \\ \alpha_i &= C \times 4 & \text{si } 30 \leq i \leq 32 \\ \alpha_i &= C \times 3 & \text{si } 33 \leq i \leq 35 \\ \alpha_i &= C \times 2 & \text{si } 36 \leq i \leq 38 \\ \alpha_i &= C \times 1 & \text{si } 39 \leq i \leq 40 \end{aligned}$$

(C est la constante
de normalisation)

T = 20

k	6	7	8	9	10	11	12	13	14
Pr [w(T)=k]	0,000	0,000	0,002	0,013	0,051	0,128	0,216	0,248	0,194
Approximation gaussienne	0,000	0,000	0,003	0,014	0,051	0,127	0,215	0,246	0,192

k	15	16	17	18	19	20
Pr [w(T)=k]	0,102	0,036	0,008	0,001	0,000	0,000
Approximation gaussienne	0,102	0,037	0,009	0,002	0,000	0,000

T = 80

k	19	20	21	22	23	24	25	26	27
Pr [w(T)=k]	0,000	0,001	0,004	0,016	0,047	0,100	0,162	0,203	0,193
Approximation gaussienne	0,000	0,001	0,005	0,017	0,047	0,100	0,161	0,201	0,193

k	28	29	30	31	32	33
Pr [w(T)=k]	0,142	0,081	0,034	0,011	0,003	0,000
Approximation gaussienne	0,142	0,081	0,035	0,012	0,003	0,001

T = 480

k	32	33	34	35	36	37	38	39	40	41
Pr [w(T)=k]	0,000	0,000	0,001	0,011	0,062	0,211	0,396	0,255	0,065	
Approximation gaussienne	0,000	0,000	0,000	0,007	0,065	0,238	0,369	0,244	0,068	0,008

k	42
Pr [w(T)=k]	
Approximation gaussienne	0,000

FIGURE 2 Suite

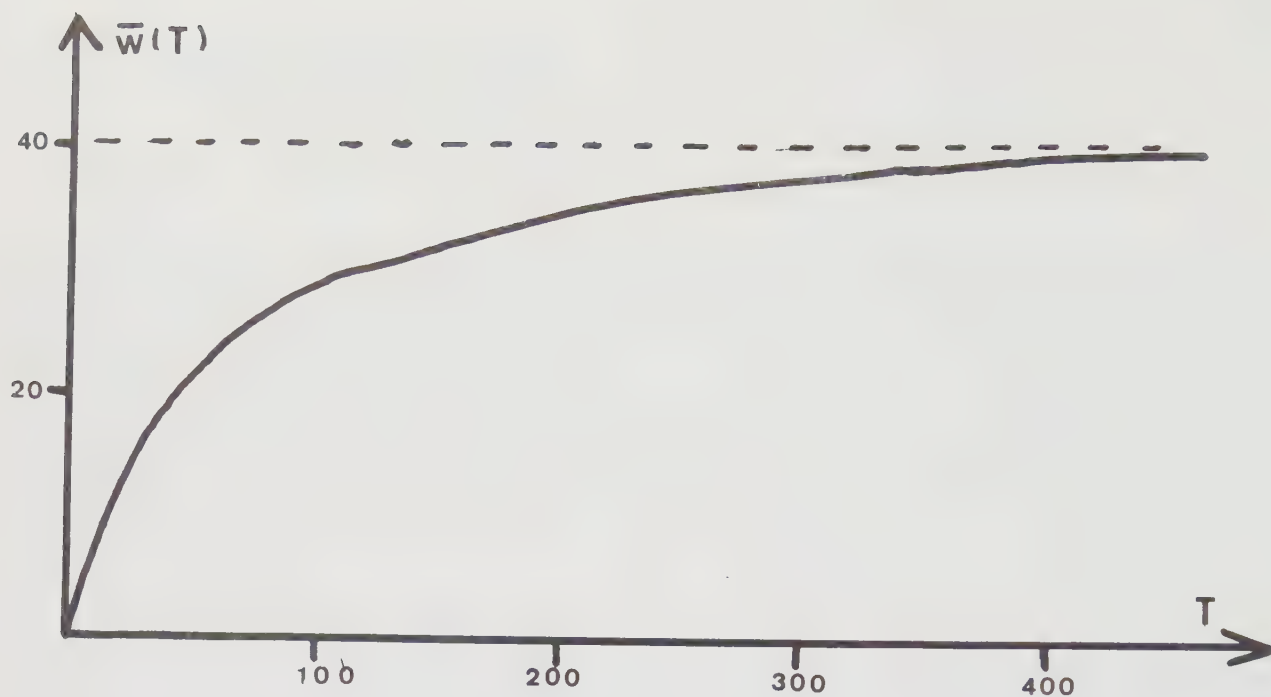


FIGURE 3

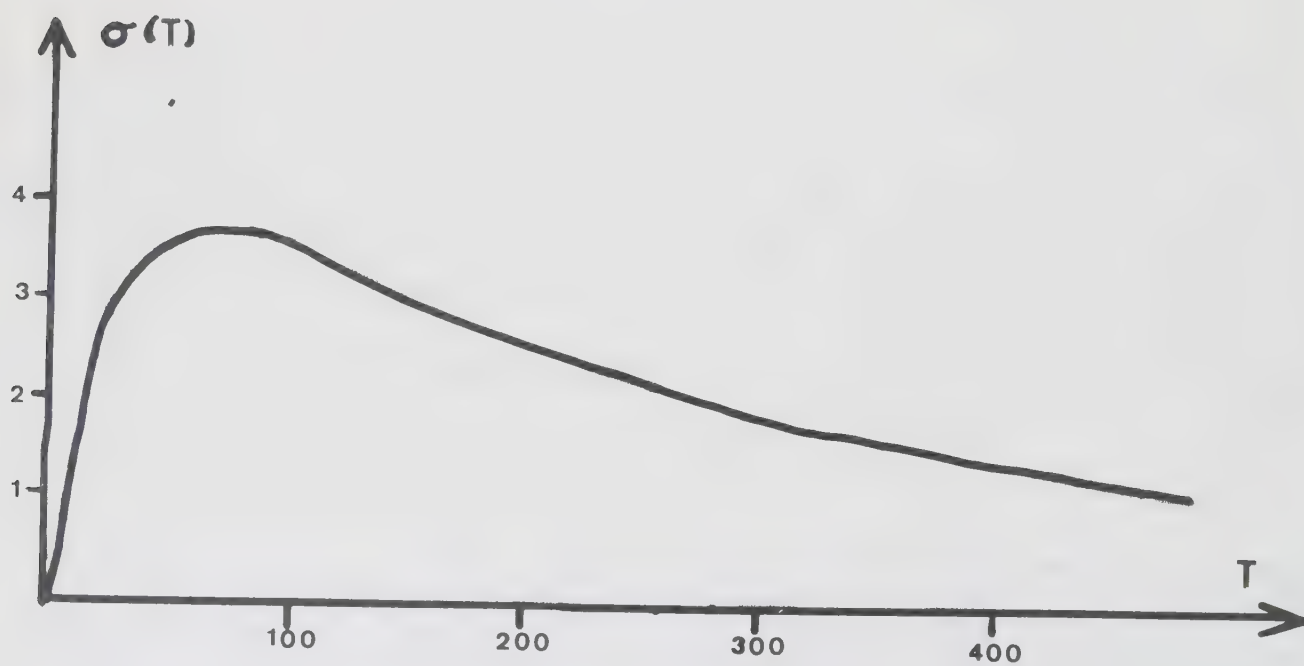


FIGURE 4

REFERENCES

- [1] DENNING, P.J. : The working set model for program behaviour
- Comm. ACM 11, 5, 323-333 (1968)

- [2] DENNING, P.J., SCHWARTZ, S.C. : Properties of the working set model
- Comm. ACM 15, 3, 191-198 (1972)

- [3] COFFMAN, E.G., RYAN, T.A. : A study of storage partitioning using
a mathematical model of locality
-Comm. ACM 15, 3, 185-190 (1972)

- [4] MORRIS, J.B. : Demand paging through utilization of working sets on
the Maniac II -Comm. ACM 15, 10, (1972)

- [5] LENFANT, J. : Evaluation sur des modèles de comportement de programme de la taille d'un ensemble de travail
- RAIRO, Revue Bleue, B2, 77-92 (1974)

- [6] HALL, M. : Combinatorial theory - Blaisdell Publishing Company,
Waltham Mass. (1967)

CHAPITRE CINQ

RESULTATS EXPERIMENTAUX

Ce chapitre est consacré à la présentation de données expérimentales relatives au comportement des programmes. Nous nous intéresserons aux caractéristiques des instructions exécutées ainsi qu'à la dispersion des cellules de mémoire qui sont référencées par des programmes réels. L'exécution interprétative d'un échantillon de programmes a permis, en particulier, de relever des chaînes de plusieurs millions de références. Nous avons utilisé cette masse de données pour mesurer la fréquence des ruptures de séquence et la taille de l'ensemble de travail ainsi que pour étudier l'influence de la taille de page ou de la taille de la mémoire physique sur le taux de défauts de page. Nous en avons également tiré des indications sur la validité de certains modèles de comportement utilisés dans les chapitres précédents.

Le calculateur utilisé pour ces expériences fut un CII 10070 [1]. La cellule fondamentale de sa mémoire centrale est le mot de 32 bits. Toute instruction du "langage machine" est représentée dans un tel mot ; la plupart des instructions opèrent sur le contenu d'un mot. Il existe quelques instructions qui référencent des double-mots, des demi-mots ou des octets ; l'utilisation de ces résolutions manque toutefois de souplesse (l'indexation est nécessaire dans les deux derniers cas cités). Ce calculateur est doté de seize registres généraux dont le contenu est, en général, identique à celui des mots 0 à 15 de la mémoire principale. Ces derniers ne sont pas à la disposition du programmeur ; ils peuvent être utilisés par les unités d'échange, ce qui explique qu'ils ne soient pas toujours une copie des registres généraux.

La mémoire centrale est divisée en pages de 512 mots. Deux dispositifs permettent d'en contrôler l'emploi. D'une part, un code d'accès (deux bits) est attaché à chaque page virtuelle dont il peut limiter ou interdire l'utilisation ; une violation de cette protection résulte en un déroutement . D'autre part, chaque page physique est protégée par un verrou (deux bits) contre toute tentative d'écriture par un programme ne disposant pas de la clé adéquate. Une dernière caractéristique du CII 10070 qui nous intéressera est l'existence de deux modes de fonctionnement : Maître/Esclave. Certaines instructions élémentaires -dites "privilégiées"- ne peuvent être exécutées qu'en mode Maître.

Il est clair que les résultats que nous allons présenter sont largement tributaires des caractéristiques du calculateur utilisé et aussi de son logiciel de base. Ils sont néanmoins comparables à ceux qui ont été obtenus avec d'autres systèmes [2 à 17].

1- L'OUTIL DE MESURE

1.1. CHOIX DE LA METHODE

Pour prélever des données telles que le numéro des instructions utilisées ou les adresses référencées durant l'exécution de programmes, plusieurs méthodes sont a priori possibles.

Moniteur matériel : Ce type de dispositif peut fort bien relever les grandeurs auxquelles nous nous intéressons et éventuellement effectuer une sélection. Mais le flot d'information produit est trop important pour que son transfert de la mémoire privée du moniteur vers une bande magnétique puisse avoir lieu en parallèle avec l'exécution du programme étudié. On ne peut donc obtenir que des échantillons portant chacun sur quelques centaines (au plus) de références consécutives. Cette méthode aurait l'avantage de la rapidité. Toutefois, même par des méthodes logicielles, la durée de cette extraction de données "brutes" n'est pas l'élément critique du coût de ces expériences : le traitement ultérieur est beaucoup plus dispendieux.

Contrôle des accès aux pages : Pour étudier le comportement d'un programme dans son espace d'adressage, on peut tirer parti des dispositifs de verrouillage liés aux pages. Une telle méthode consiste à interdire l'accès à toutes les pages autres que celle qui est présentement utilisée. Lorsqu'une autre page est référencée, le déroutement qui se produit réveille un programme d'extraction des données. Le manque de souplesse des instructions de contrôle des accès et l'absence de dispositif matériel pour analyser les déroutements "violation de protection" rendent cette méthode peu efficace dans le cas du calculateur CII 10070. De toutes façons, on ne pourrait pas utiliser les chaînes de références ainsi relevées pour étudier des hiérarchies de mémoire où la taille de page serait inférieure à celle du calculateur utilisé.

Cet inconvénient est assez gênant ; le prélèvement de chaînes de références par verrouillage d'accès est pratique dans les systèmes segmentés [13].

Exécution interprétative : C'est la méthode que nous avons adoptée.

Un programme, l'"extracteur", provoque l'exécution, instruction par instruction, du programme à étudier. Il relève un certain nombre de données sur la prochaine instruction (en particulier son code opératoire et les adresses des mots de mémoire auxquels elle accédera) puis en provoque l'exécution par un EXU. Nous allons décrire plus en détails cet outil de mesure.

1.2. FONCTIONS DE L'EXTRACTEUR

L'extracteur est destiné à l'étude de programmes usagers c'est-à-dire de processeurs ou de programmes écrits par des utilisateurs du calculateur. Les sous-programmes de bibliothèque qui ont été liés à un programme usager sont considérés comme en étant partie intégrante. En revanche, nous ne nous intéressons pas au comportement des procédures-système qui peuvent être appelées par le programme usager. Les données fournies à l'extracteur sont donc :

- Le programme usager à étudier, sous forme d'un module de chargement exécutable. L'adresse basse d'implantation en mémoire (BIAS) doit être supérieure à la valeur standard afin de ménager un emplacement pour l'extracteur lui-même.

- Les fichiers requis par le programme usager.

Si le programme à étudier est trop long pour que toutes ses instructions puissent être interprétées, l'extracteur peut analyser soit les premières références, soit plusieurs échantillons prélevés à intervalles de temps réguliers.

Dans le second cas, à la fin de l'interprétation d'un échantillon, l'extracteur initialise une horloge et passe le contrôle au programme usager étudié. Celui-ci s'exécute -sans interprétation- jusqu'à ce que l'horloge atteigne le zéro. L'extracteur se met alors à analyser un nouvel échantillon.

De l'analyse d'une instruction du programme usager, on retient les informations suivantes :

a) Les adresses de tous les mots auxquels l'instruction accède sont relevées, l'ordre des références étant préservé. Ainsi pour un chargement de mot en registre avec adressage indirect, trois adresses sont mémorisées :

- celle de l'instruction
- celle du pointeur
- celle de l'opérande.

Toutefois, les adresses inférieures ou égales à 15 ne sont pas retenues car elles repèrent des registres généraux. Outre l'adresse, l'extracteur relève la cause de l'accès à la mémoire (recherche d'une instruction/recherche d'une donnée) et, le cas échéant, le mode d'utilisation d'un mot contenant une donnée (lecture/écriture).

b) Si l'instruction du programme usager est un appel au superviseur (CALL), ses paramètres sont relevés. En particulier, lors d'un appel de procédure d'entrée/sortie, l'emplacement et la taille du tampon sont mémorisés.

c) L'extracteur comptabilise

- les occurrences de chaque code opératoire avec ventilation selon le type de l'adressage (direct ou indirect, indexé ou non).
- les ruptures de séquence.

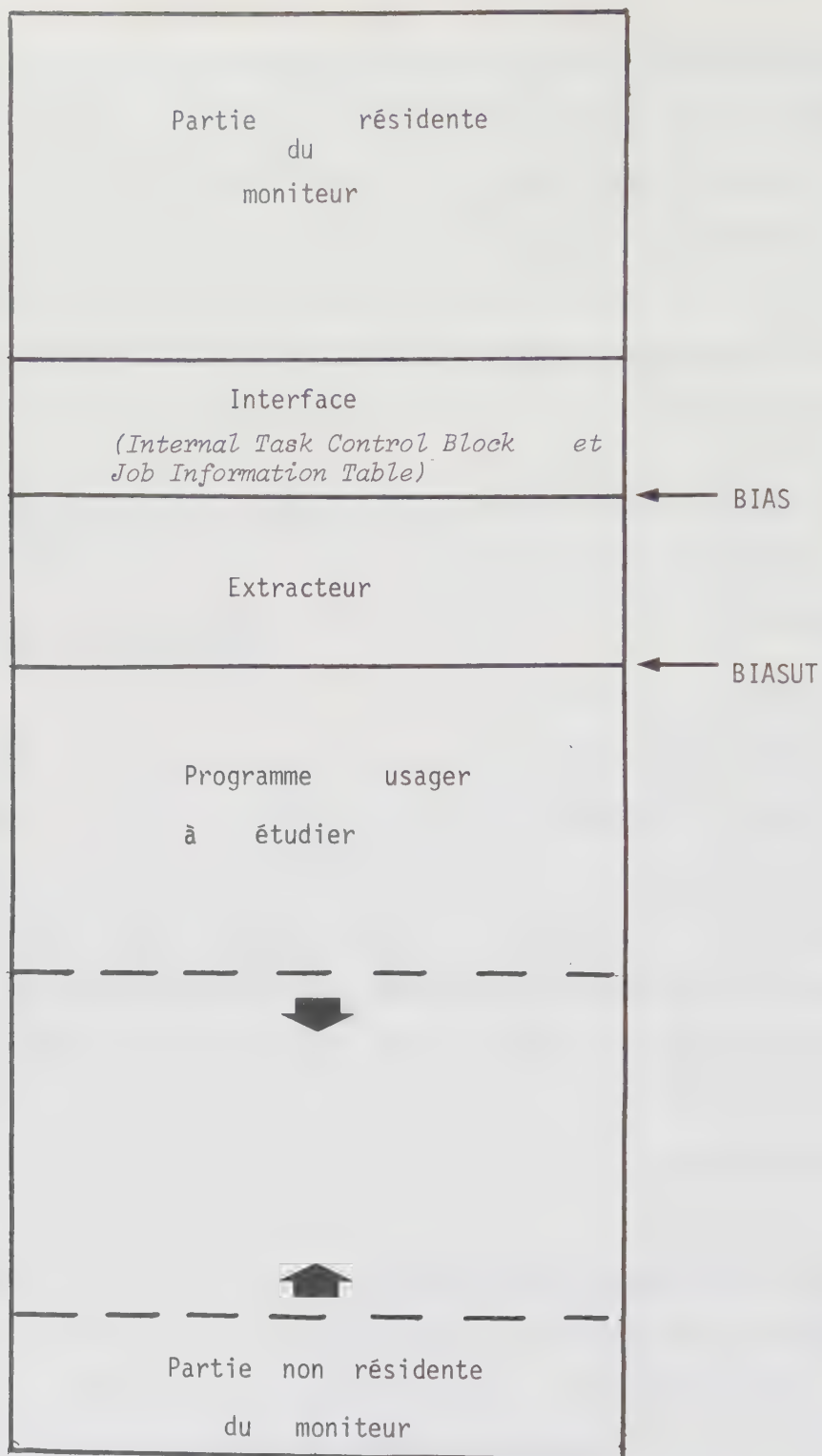


FIGURE 1

1.3. DESCRIPTION DE L'EXTRACTEUR

1.3.1. OCCUPATION DE LA MEMOIRE

Le partage de la mémoire est conforme à la figure 1. La coexistence dans la zone utilisateur de l'extracteur et du programme à étudier a posé quelques problèmes, d'autant que le second qui est chargé comme un segment de données devient de temps en temps le programme actif. Ainsi

- Le moniteur ne possède qu'une pile de sauvegarde (placée juste après l'adresse BIAS) qu'il utilise aussi bien pour l'extracteur que pour le programme usager.

- Pour les protections d'accès, le programme usager à étudier est considéré comme zone de données et non comme module exécutable.

1.3.2. LES MODULES DE L'EXTRACTEUR

L'extracteur comporte deux parties principales. La première est un chargeur qui remplace celui du moniteur SIRIS 7 tout en permettant un contrôle de la part de l'extracteur. Ce module

- gère des tables décrivant les segments du programme à étudier et la structure de l'arbre de recouvrement ;
- charge en mémoire la racine du programme usager ;
- organise les transferts de segments entre la mémoire principale et le disque ;
- positionne les codes d'accès aux pages.

La seconde partie de l'extracteur est le module "interpréteur". C'est lui qui relève les références exécutées par chaque instruction ainsi que quelques informations annexes. Comme nous l'avons déjà signalé, il commande ensuite l'exécution de l'instruction simulée (par EXU, sauf dans le cas des instructions de branchement).

1.3.3. LES RELATIONS DE L'EXTRACTEUR ET DU MONITEUR

L'extracteur et le moniteur sont amenés à coopérer par les déroutements et les appels de procédures-système. Les déroutements sont pris en charge par l'extracteur (grâce à la procédure système M : TRAP). Lorsqu'une instruction du programme étudié provoque un déroutement, l'extracteur consulte une de ses tables pour voir si un appel superviseur M : TRAP correspondant à la cause du déroutement a été interprété. Dans l'affirmative, il donne le contrôle au programme étudié ; la séquence de récupération prévue dans ce dernier est alors interprétée. Sinon le déroutement correspond à une erreur dans le programme usager, ce que signale l'extracteur.

Lorsqu'une instruction analysée est un appel au moniteur (CALL) l'extracteur en relève les paramètres. Une alternative se présente alors. Si l'appel a pour but une prise en charge de déroutement (M : TRAP , M : RETURN) ou un chargement de segment (M : SEGLD), l'action requise est accomplie par l'extracteur, ce qui est en accord avec l'organisation décrite ci-dessus. Dans les autres cas (par exemple : appel M : READ ou M : WRITE) c'est au moniteur d'exécuter la procédure adéquate. L'instruction CALL du programme étudiée est alors recopiée dans l'extracteur et c'est sur cette copie que porte l'EXU. La raison d'être de cet artifice est que, après l'exécution de la procédure système, le contrôle doit être rendu à l'extracteur et non au programme étudié.

2- L'ECHANTILLON DE PROGRAMMES

Les programmes étudiés ont été prélevés dans les bacs soumis au calculateur CII 10070 du Centre Interuniversitaire de Calcul de Bretagne. Si l'on excepte les programmes d'étudiants en Informatique, tous les programmes ainsi obtenus étaient écrits en Fortran. Les mesures rapportées dans ce chapitre ont été effectuées sur dix programmes que nous décrivons ci-dessous. Les indications entre parenthèses sont respectivement le langage source, le nombre de références analysées et la taille de l'espace virtuel du programme *logper* (ce terme a été précisé au paragraphe 1.2.) en pages de 512 mots.

Programme A (Fortran, 3 998 000 références, 57 pages)

Résolution par la méthode des éléments finis d'un système d'équations différentielles déterminant la distribution des vitesses sur un profil d'avion. La plus grande partie de la chaîne de références a été prélevée pendant l'exécution d'un sous-programme effectuant une triangulation.

Programme B (Fortran, 2 201 000 références, 10 pages)

Etude, sous plusieurs hypothèses, de l'évolution dans le temps d'un troupeau de bovins.

Programme C (LP 70, 4 000 000 références, 20 pages)

Etude en simulation d'une hiérarchie de mémoire à trois niveaux : arborescence - mémoire principale - tambour. Les adresses référencées sont simulées selon le modèle LPU de comportement.

Programme D (Fortran, 3 998 000 références, 37 pages)

Dépouillement d'une enquête sur la circulation automobile dans le centre de Rennes. Le programme effectue un contrôle des données de l'enquête (par exemple : détection des véhicules dont l'entrée dans le centre de la ville a été repérée sans que la sortie le soit) puis évalue la durée probable du stationnement et calcule l'évolution horaire du trafic. Le fichier de données est sur cartes.

Programme E (Fortran, 2 265 000 références, 72 pages)

Optimisation de la synchronisation des feux tricolores dans un réseau de rues.

Programme F (Fortran et LP 70, 3 997 000 références, 42 pages)

Calcul de plusieurs distributions de probabilité. L'une d'elles nécessite une sommation de termes indexés par l'ensemble des parties d'un ensemble (utilisation d'une pile).

Programme G (LP 70, 2 234 000 références, 44 pages)

Ce programme est le compilateur LP 70. Il a été observé durant toute la traduction d'un programme d'environ 650 cartes.

Programme H (Algol, 554 000 références, 9 pages)

Exercice sur la compilation des expressions arithmétiques.

Programme I (Fortran, 1 308 000 références, 8 pages)

Analyse de variance. Ce programme comporte une boucle portant sur l'ensemble du code.

Programme J (Fortran, 2 257 000 références, 11 pages)

Approximation de plusieurs ensembles de points par des polynômes selon le critère des moindres carrés.

Remarquons que l'influence du choix du langage source est difficile à cerner. En effet, les sous-programmes de bibliothèque font partie intégrante des programmes usagers que prend en compte l'extracteur. Or ces sous-programmes ont eux-mêmes été écrits dans plusieurs langages, en particulier, en Metasymbol (langage d'assemblage). Par ailleurs, Fortran et Algol utilisent la même collection de sous-programmes d'entrée/sortie : il y a là une cause d'unification des comportements qui ne peut être négligée.

L'importance des sous-programmes de service (en particulier, de ceux qui préparent les entrées/sorties) est d'ailleurs une caractéristique programmes de l'échantillon. Si les instructions de ces sous-programmes ne représentent qu'environ 6% et 23% du nombre total d'instructions exécutées par les programmes C et A, ce pourcentage est de 78% pour le programme D, de 80% pour F et même de 93% pour B ! Cette répartition du temps de calcul que les travaux publiés sur le comportement des programmes ne prennent généralement pas en compte, a deux causes. D'une part, la quantité d'informations dont l'impression est demandée est souvent importante, voire excessive (résultats intermédiaires à chaque itération pour mettre au point le programme, ...). D'autre part, l'interprétation des FORMAT est très coûteuse, d'autant que les possibilités offertes par le Fortran CII vont très au-delà des normes ou même des spécifications des autres systèmes. Le nombre moyen d'instructions exécutées par les sous-programmes de bibliothèque (à l'exclusion des instructions exécutées par le moniteur) à l'occasion de l'impression d'une ligne ou de la lecture d'une carte est compris entre 2 500 et 5 000 (ce nombre est une moyenne sur l'ensemble des enregistrements lus ou imprimés par un programme). Pour tenir compte de cet effet, lorsque nous présenterons des résultats sur un échantillon réduit de programmes, nous y inclurons les programmes A, B, C et D.

3- UTILISATION DES INSTRUCTIONS

3.1. FREQUENCE D'UTILISATION DES DIVERSES INSTRUCTIONS

Pour faciliter la lecture des résultats, les instructions du calculateur CII 10070 ont été rassemblées en dix groupes qui sont décrits dans l'annexe. En particulier, les instructions de comparaison ont pour effet de positionner le code de condition après comparaison de leurs opérandes. Elles sont donc normalement suivies d'une instruction de branchement qui teste le code de condition. Ce dernier est affecté par de nombreuses autres instructions, notamment par les chargements et les opérations arithmétiques. C'est ainsi que pour la comparaison du contenu d'un registre R avec 0, le compilateur Fortran génère une addition immédiate :

AI,R	0	(durée : 1,2 μ s)
------	---	-----------------------

au lieu d'une comparaison immédiate :

CI,R	0	(durée : 1,8 μ s)
------	---	-----------------------

Comme il aurait été extrêmement complexe de détecter tous les cas où une instruction est ainsi utilisée pour un de ses effets secondaires, nous avons pris le parti de ne jamais considérer les additions immédiates comme instructions de comparaison : ceci ne doit pas affecter beaucoup les résultats.

La figure 2 représente la fréquence, en pourcentage avec laquelle les instructions des différents groupes sont exécutées. La part des instructions de branchement et de comparaison est considérable, respectivement :

42,2%	45,4%	62,2%	50,6%	55,0%	41,3%	54,4%
-------	-------	-------	-------	-------	-------	-------

Programme Fonction	A	E	C	D	E	G	H
Chargement/ Rangement	29,6	27,7	20,4	30,5	25,6	33,9	26,2
Branchement	24,0	33,3	38,4	35,0	38,6	31,2	38,0
Comparaison	18,2	12,1	23,8	15,6	16,4	10,1	16,4
Arithmétique entière	18,1	17,1	16,4	11,3	11,0	15,0	11,0
dont AI	9,2	11,0	13,9	8,0	6,8	5,5	7,9
Arithmétique réelle	8,1	3,8	0,2	0	2,1	0	0,00
Arithmétique en base 10	0	0	0	0	0	0	0
Logique	0,6	1,3	0,1	3,3	1,8	3,5	2,8
Appel au moniteur	0,01	0,03	0,003	0,01	0,01	0,04	0,1
Décalage	0,9	3,2	0,5	2,5	3,1	2,3	4,0
Traitement de chaînes	0	0	0	0	0	0,6	0
Divers	0,3	1,5	0,1	1,8	1,4	3,3	1,6

Fréquence de chaque fonction

FIGURE 2

	Programme A		Programme B		Programme C	
1	LW	17,4 %	BCR	15,2 %	BCS	21,9 %
2	CW	15,7 %	AI	11,0 %	BCR	15,7 %
3	BCR	11,2 %	STW	10,1 %	AI	13,9 %
4	BCS	9,6 %	LW	6,7 %	CW	13,4 %
5	STW	9,2 %	CW	6,4 %	CI	9,8 %
6	AI	9,2 %	BCS	6,3 %	LW	8,0 %
7	FSS	8,0 %	BDR	6,2 %	STW	4,2 %
8	LAW	7,9 %	STB	5,1 %	LB	2,9 %
9	BDR	2,0 %	BAL	4,7 %	STB	2,8 %
10	CI	1,3 %	CI	3,4 %	LI	1,7 %
11	BAL	1,3 %	S	3,2 %		
12	STB	1,1 %	LI	3,1 %		
13			FMS	2,0 %		
14			FAS	1,8 %		
15			AW	1,8 %		
16			LB	1,5 %		
17			CB	1,5 %		
18			LCFI	1,4 %		
19			MW	1,4 %		

FIGURE 3

AI	: Addition immédiate (1,2)
AW	: Addition au contenu d'un registre du contenu d'un mot (1,8)
BAL	: Branchement avec préservation de l'adresse dans un registre (2,2)
BCR	: Branchement si le produit logique d'un masque (passé en paramètre) et du code condition est nul (1,0 ou 1,9)
BCS	: A la différence de BCR, le branchement a lieu lorsque le produit est non nul (1,0 ou 1,9)
BDR	: Décrémentation d'un registre et branchement si la valeur résultante est négative (1,5 ou 2,4)
CB	: Comparaison de deux octets dont l'un est dans un registre (1,8)
CI	: Comparaison immédiate (1,8)
CW	: Comparaison du contenu d'un mot et du contenu d'un registre (1,8)
FAS	: Addition réelle en simple précision (4,0)
FMS	: Multiplication réelle en simple précision (5,4)
FSS	: Soustraction réelle en simple précision (3,3)
LAW	: Chargement dans un registre de la valeur absolue d'un mot (1,8)
LB	: Chargement d'un octet dans un registre (1,8)
LCFI	: Forçage du code condition (1,2)
LI	: Chargement immédiat (1,2)
LW	: Chargement dans un registre du contenu d'un mot (1,8)
MW	: Multiplication entière (4,9)
S	: Décalage (2,3 à 13,4)
STB	: Rangement d'un octet (2,7)
STW	: Rangement du contenu d'un registre (2,4)

Signification des mnémoniques et durée de l'exécution
en microsecondes

FIGURE 3 (suite)

Le pourcentage particulièrement élevé pour le programme C est une conséquence de la méthode de génération d'adresses et de la mise à jour de la liste LRU : ces deux opérations nécessitant de nombreux tests sont répétées à chaque simulation d'une instruction (ce programme, rappelons le est un simulateur de système informatique à mémoire hiérarchisée).

On remarquera par ailleurs que pour chacun des programmes, les instructions de rangement/chargement, branchement et comparaison représentent entre 80 et 83 pour cent de toutes les instructions exécutées (sauf pour G : 75%). Parmi les instructions arithmétiques, nous avons distingué l'addition immédiate (AI) qui est utilisée pour incrémenter des variables à la demande explicite du programmeur, mais aussi pour le calcul de déplacements dans les tableaux ou pour la génération de certaines boucles. La relative rareté des instructions arithmétiques a pour conséquence que le nombre des multiplications (par exemple) est un critère de complexité des algorithmes qui doit être utilisé avec circonspection dans la pratique.

Pour trois programmes de l'échantillon, nous avons classé les instructions utilisées avec une fréquence supérieure à 1% (voir figure 3). Malgré des disparités dans le classement, les sept instructions AI, BCR, BCS, CI, CW, LW, STW figurent parmi les dix instructions les plus utilisées par chacun des trois programmes.

3.2. CALCUL DES ADRESSES

Les instructions du calculateur CII 10070 sont, pour la plupart, à deux opérandes. L'un de ces opérandes est généralement le contenu d'un registre. L'autre est désigné par la zone dite *Adresse* de l'instruction. La figure 4 montre qu'environ 20% des instructions exécutées interprètent le contenu de cette zone *Adresse* comme une valeur. Les autres l'interprètent effectivement comme une adresse.

	A	B	C	D	E	F	G	H	I	J
Valeur										
("Adresse immédiate")	12,1	19,4	25,3	18,4	18,7	18,8	21,4	21,0	21,5	20,9
Adresse d'octet	2,4	8,1	6,2	6,8	8,1	6,7	12,7	6,9	10,7	7,6
Adresse de demi-mot	0,02	0,02	0,7	0,02	0,006	0,03	2,3	0,05	0,06	0,05
Adresse de mot	85,1	71,0	67,5	73,8	71,9	64,5	65,6	70,6	66,0	66,4
Adresse de double mot	0,4	1,5	0,3	1,0	1,4	9,9	0,2	1,5	1,8	5,0

Utilisation de la zone *Adresse*

FIGURE 4

	A	B	C	D	E
Indirection	8,7	3,5	1,7	2,0	5,4
Indexation	12,2	19,2	21,4	14,0	14,7

	F	G	H	I	J
Indirection	4,2	6,3	3,0	3,4	6,0
Indexation	14,0	17,4	12,3	15,6	14,4

FIGURE 5

Le calculateur CII 10070 est une machine à mots ce qui apparaît nettement sur la figure 4. Les instructions sur demi-mot ne représentent que quelques dix-millièmes du total, bien que toutes les opérations arithmétiques soient disponibles dans ce format. Cela s'explique sans doute par le fait que la moitié droite d'un mot ne peut être repérée que par indexation, ce qui manque de souplesse.

Pour chaque programme, la figure 5 montre le pourcentage des instructions ayant requis des mécanismes d'indirection ou d'indexation.

4 - LES RUPTURES DE SEQUENCE -

Nous avons vu au paragraphe 3.1. qu'une instruction sur trois est une instruction de branchement. La figure 6 montre quel est leur effet. Le pourcentage des ruptures de séquence a été évalué d'une part par rapport au nombre total d'instructions exécutées, d'autre part par rapport au nombre d'instructions de branchement. Le nombre moyen d'instructions exécutées entre deux ruptures de séquence est de 4. La fraction des instructions de branchement qui produisent une rupture de séquence (comprise entre 65% et 82%) peut être assez faible si l'on tient compte des GOTO, appels de sous-programmes et des boucles (DO, REPEAT, WHILE). En fait, il est beaucoup de boucles qui ne sont pas répétées un nombre élevé de fois. Un indice en est le nombre de ruptures de séquence provoquées par l'instruction BIR (Branch and Increment Register) qui est générée par le compilateur Fortran pour traduire les boucles DO lorsque les bornes sont des valeurs (et non des variables) : 73% pour les programmes D et E (soit 4 itérations en moyenne), 85% pour le programme A. Le pourcentage est néanmoins de 97% pour B.

La fréquence des appels à des sous-programmes nous est fournie par la considération des occurrences de l'instruction BAL (*Branch and Link* : branchement avec sauvegarde de l'adresse de rupture). La figure 7 montre la fréquence de BAL dans la suite de toutes les

	A	B	C	D	E
Instructions de branchement	24,0	33,3	38,4	35,0	38,6
Ruptures de séquence rapportées au nombre total d'instructions	19,6	25,1	25,0	24,9	25,3
Ruptures de séquence rapportées au nombre d'instructions de branchement	82	76	65	71	66

	F	G	H	I	J
Instructions de branchement	35,2	31,2	37,9	37,4	33,8
Ruptures de séquence rapportées au nombre total d'instructions	25,1	23,4	25,5	26,4	23,9
Ruptures de séquence rapportées au nombre d'instructions de branchement	71	75	67	76	70

FIGURE 6

Programme	A	B	C	D	E	F	G	H
Fréquence de BAL	1,3%	4,7%	0,6%	4,2%	4,7%	5,6%	1,9%	4,5%

FIGURE 7

instructions exécutées. Certains de ces pourcentages peuvent paraître très élevés. La faute en incombe principalement aux sous-programmes de bibliothèque qui traitent les entrées/sorties. On peut s'en assurer en considérant les programmes B, D et H qui n'utilisent pas de fonction intrinsèque ni de sous-programme écrit par l'utilisateur ou placé dans une bibliothèque autre que la bibliothèque "de servitude". Notons toutefois que la plupart des sous-programmes de service ne procèdent pas à des sauvegardes de registre ni à des déplacements de paramètres : le coût de cette multitude d'appels n'est donc pas forcément excessif.

5- PROPRIETES DES CHAINES DE REFERENCES

Le principal produit de l'extracteur est la suite des adresses des mots auxquels le programme usager a accédé durant la période d'observation. Adresses d'instructions et adresses de données s'y succèdent dans l'ordre chronologique : les références à des instructions

	A	B	C	D	E	F	G	H	I	J
Pourcentage d'instructions	62	70	77	70	72	72	70	73	74	72

FIGURE 8

interviennent pour environ 72% du total (voir figure 8). Les pointeurs étant comptabilisés comme données, le faible pourcentage d'instructions du programme A s'explique par la fréquence élevée des indirections (voir figure 5) ; cette dernière est elle-même due à l'usage répété d'un paramètre d'un sous-programme.

5.1. DISTRIBUTION DES REFERENCES AUX PAGES

Nous allons désormais supposer que les espaces d'adresses sont divisés en pages de taille TPAGE. Pour un programme donné, nous désignerons par n le nombre de ces pages et par β_i ($1 \leq i \leq n$) la fréquence relative avec laquelle la page i a été référencée. Les nombres $\tilde{\beta}_i$ sont des estimateurs des probabilités β_i (chapitre 1, paragraphe 2.2). Nous supposons la numérotation des pages telle que $\tilde{\beta}_i$ soit une fonction décroissante de i . L'importance relative des pages (en temps d'exécution) varie considérablement. Ce fait est attesté par l'étude de la fonction

$$P(x) = \sum_{i=1}^x \tilde{\beta}_i \quad (0 \leq x \leq n)$$

Nous avons évalué cette fonction sur la suite de toutes les références ainsi que sur la sous-suite des adresses d'instructions et la sous-suite des adresses de données.

Les résultats obtenus pour les programmes C et D sont représentés sur les figures 9 et 10. A différentes tailles de page correspondent évidemment différentes courbes (figure 11). La concentration des références est surprenante : plus de 80 pour cent d'entre elles portent sur moins de 15 pour cent des pages. De plus, la dispersion des adresses d'instruction est bien moindre que celle des adresses de donnée, ce que va confirmer le paragraphe ci-dessous.

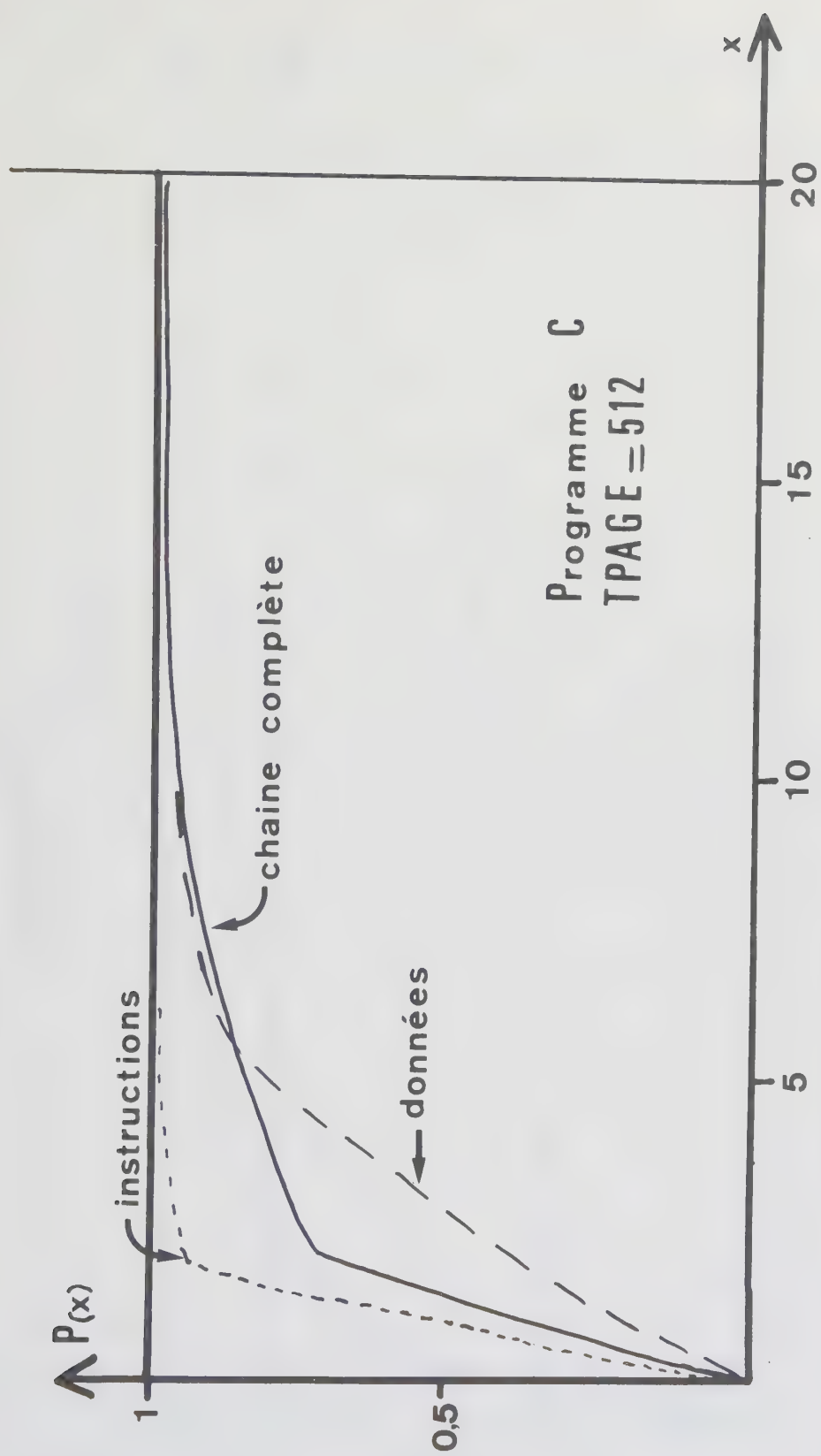


FIGURE 9

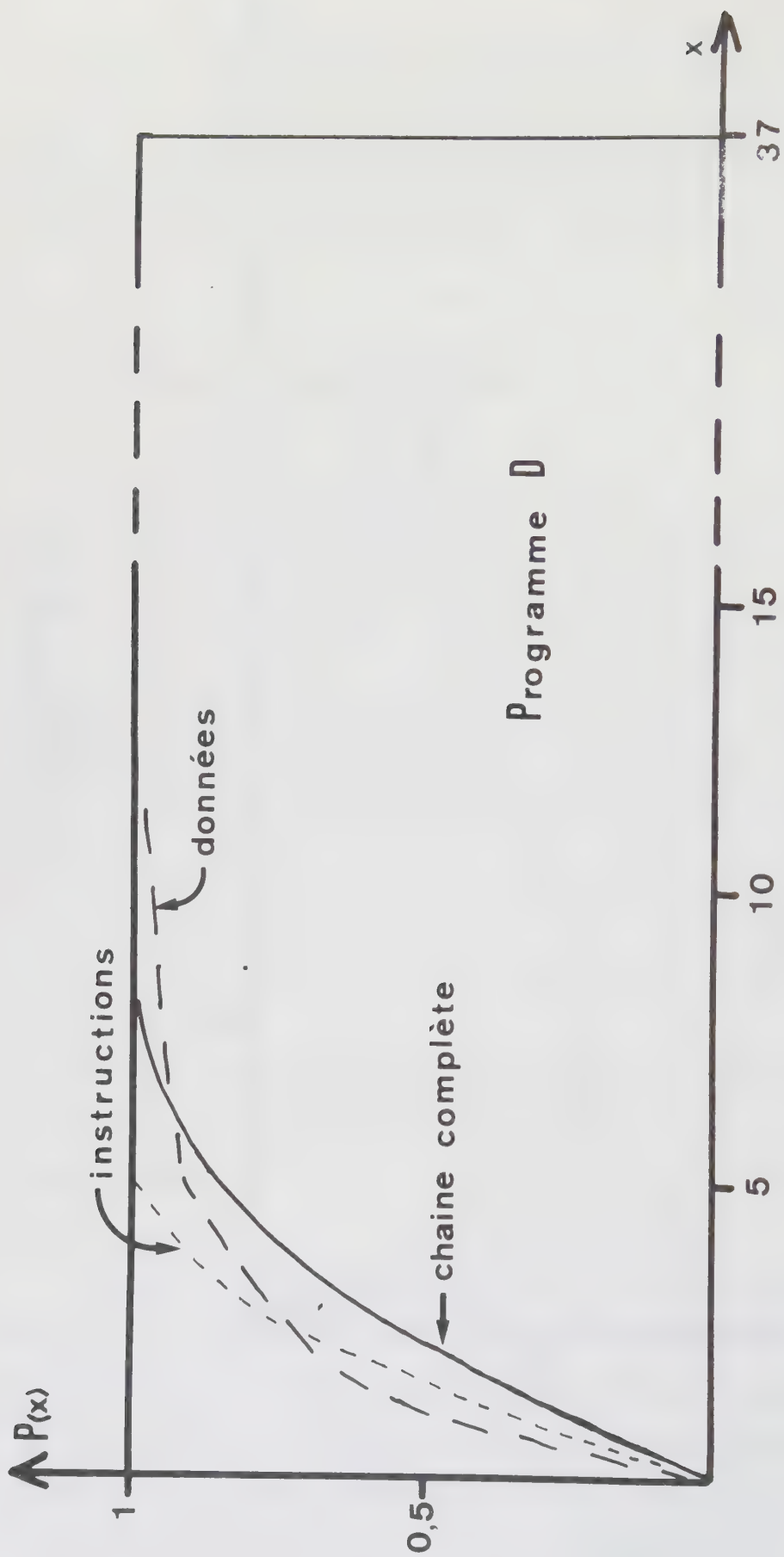


FIGURE 10

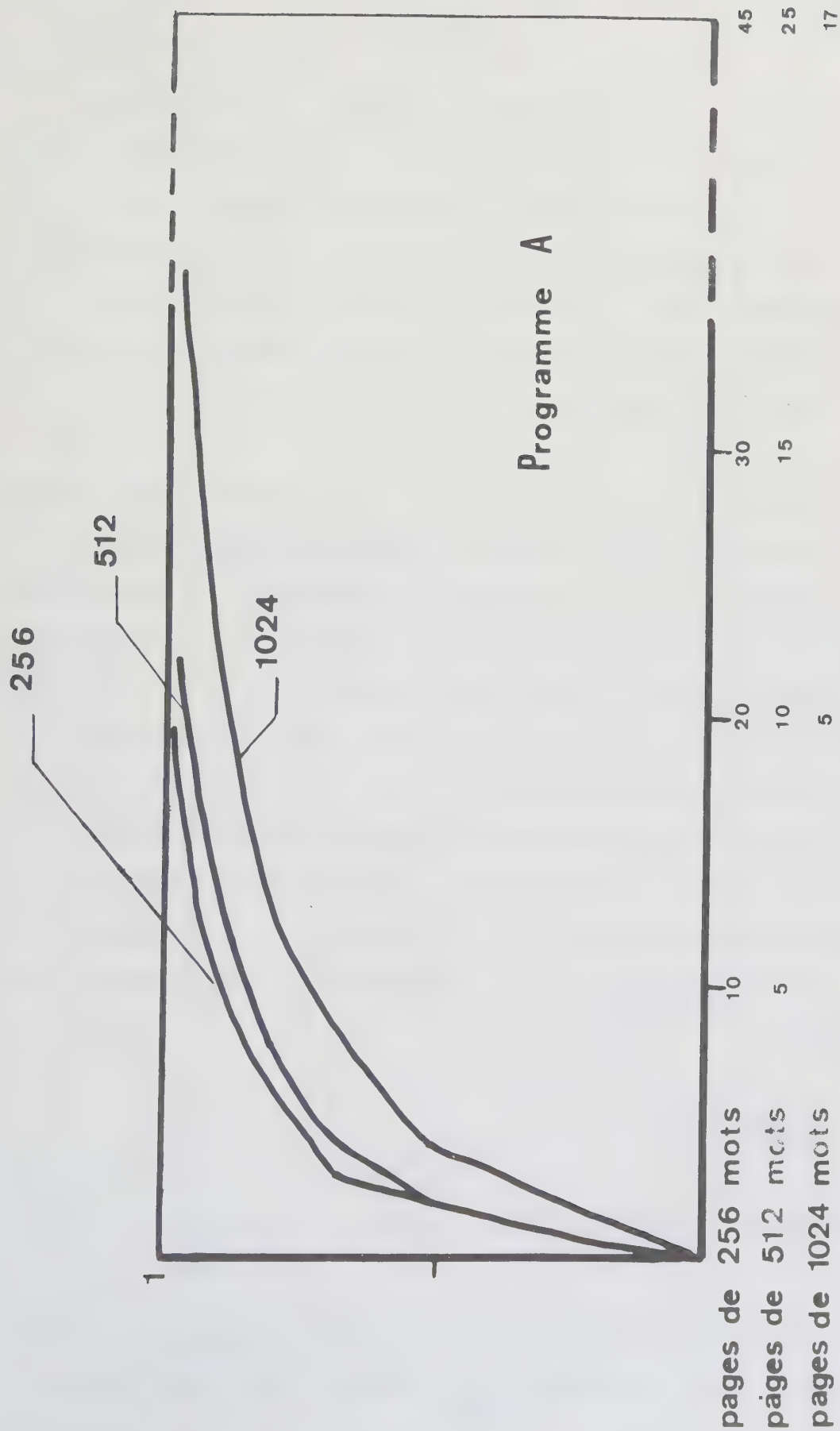


FIGURE 11

5.2. DISTRIBUTION DES DISTANCES DANS LA LISTE LRU

Avec une fréquence que nous désignerons par $\hat{\alpha}_i$ ($1 \leq i \leq n$), la page référencée est celle qui se trouvait à la distance i dans la liste LRU. Deux exemples de la distribution ($\hat{\alpha}_i$) sont présentés sur la figure 12. L'espace d'adresses du programme A est divisé en 57 pages dont 25 seulement ont été référencées durant la période d'observation (ces deux nombres sont respectivement 36 et 17 pour le programme E dont toute l'exécution a été interprétée). On notera la décroissance rapide (mais non monotone) de $\hat{\alpha}_i$.

Pour apprécier l'éparpillement des adresses d'instructions et des adresses de données, nous avons appliqué à chacune des deux sous-suites la procédure qui avait été employée pour la suite complète de références. Pour chaque instruction exécutée, c'est donc dans la liste LRU des adresses d'instructions -et non dans la liste LRU "globale" - qu'est évaluée la distance. Il en est de même pour les données. Les mesures relatives à ces deux sous-chaînes sont également illustrées par la figure 12. On remarque que deux instructions exécutées l'une après l'autre sont généralement logées dans la même page : la fréquence de cette coïncidence oscille entre 0,95 et 0,998 pour les programmes de l'échantillon et les tailles de page considérées (256, 512 ou 1024 mots). En revanche, les adresses de données sont beaucoup plus dispersées (voir, en particulier, le cas du programme A).

6- TAUX DE DEFAULTS DE PAGE

6.1. INFLUENCE DE LA TAILLE DES PAGES

Trois algorithmes de remplacement de pages ont été pris en considération : FIFO et LRU pour leur intérêt pratique, OPT parce qu'il peut servir d'étalon

Distance	Chaîne complète	Instruc- tions	Données
1	0,4205	0,8912	0,2859
2	0,3612	0,0134	0,1434
3	0,0578	0,0031	0,1950
4	0,1291	0,0005	0,3187
5	0,0212	0,0010	0,0450
6	0,0046	0,0005	0,0031
7	0,0018	0,0001	0,0038
8	0,0015	= 0	0,0020
9	0,0012		0,0005
10	0,0001		0,0010
11	0,0001		0,0003
12	0,0002		0,0004
13	0,0003		0,0002
14	0,0001		0,0003
15	0,0001		0,0002
> 15	$< 10^{-6}$		$< 10^{-6}$

TPAGE = 512

PROGRAMME A

Chaîne complète	Instruc- tions	Données
0,5657	0,9863	0,7136
0,3631	0,0104	0,1435
0,4035	0,0017	0,0888
0,0233	0,0016	0,0403
0,0026	$< 10^{-6}$	0,0060
0,0009		0,0043
0,0001		0,0005
0,0001		0,0007
0,0006		0,0022
$< 10^{-4}$		0,0002

TPAGE = 1024

PROGRAMME E

FIGURE 12

PROGRAMME A

TPAGE	Pages en M.C	OPT	LRU	FIFO
32	192	0,00007	0,00007	0,00007
64	96	0,00004	0,00004	0,00004
128	48	0,00002	0,00003	0,00003
256	24	0,00001	0,00002	0,00003
512	12	0,00017	0,00051	0,00070
1024	6	0,00245	0,00435	0,00596
2048	3		0,02894	0,04325

Taille du programme : 25 pages de 512 mots

PROGRAMME D

TPAGE	Pages en M.C	OPT	LRU	FIFO
32	192	0,00007	0,00007	0,00008
64	96	0,00004	0,00004	0,00004
128	48	0,00002	0,00003	0,00004
256	24	0,00004	0,00010	0,00004
512	12	0,00019	0,00050	0,00111
1024	6	0,00077	0,00089	0,00112
2048	3	0,00259	0,00280	0,00447

Taille du programme : 28 pages de 512 mots

Taux de défauts de page

FIGURE 13 (a)

PROGRAMME B

TPAGE	Pages en M.C	OPT	LRU	FIFO
8	256	0,00169	0,00451	0,00506
16	128	0,00125	0,00250	0,00278
32	64	0,00104	0,00160	0,00202
64	32	0,00120	0,00285	0,00418
128	16	0,00192	0,00353	0,00468
256	8	0,00514	0,00856	0,01109
512	4	0,02248	0,03263	0,04480
1024	2		0,07923	0,10298

Taille du programme : 10 pages de 512 mots

PROGRAMME C

TPAGE	Pages en M.C	OPT	LRU	FIFO
16	192	0,00009	0,00009	0,00009
32	96	0,00005	0,00005	0,00007
64	48	0,00003	0,00003	0,00004
128	24	0,00036	0,00089	0,00175
256	12	0,00230	0,00568	0,00812
512	6	0,00631	0,00927	0,01549
1024	3		0,02758	0,03764

Taille du programme : 20 pages de 512 mots

Taux de défauts de page

FIGURE 13 (b)

pour apprécier les autres algorithmes. Pour mesurer le taux de défauts de page d'OPT, nous avons utilisé un algorithme qui traite séquentiellement la chaîne de références (algorithme MIN : [22]). Pour un algorithme donné, le taux de défauts de page moyen F pendant l'exécution d'un programme est la fraction des adresses de la chaîne qui désignent une page ne se trouvant pas en mémoire à l'instant où la référence est exécutée ; nous ne distinguons pas les références aux données des références aux instructions.

A chacun des programmes de l'échantillon, nous avons attribué une zone de mémoire centrale égale à moins de la moitié de son espace virtuel. La taille (en mots) de cette zone étant maintenue constante, pour les trois ARP mentionnés ci-dessus, nous avons évalué le taux de défauts de page F en fonction de la taille de page TPAGE. Il est apparu que la courbe représentative de $F(\text{TPAGE})$ a la même allure dans tous les cas :

- . Il existe une taille de page t_0 telle que le taux de défauts de page décroît lentement tant que $\text{TPAGE} \leq t_0$.
- . Le taux de défauts de page croît rapidement lorsque TPAGE devient supérieur à t_0 .
- . Pour une chaîne de références et un ARP donnés, la fonction $F(\text{TPAGE})$ ne possède pas d'autre extrémum que t_0 . La valeur de t_0 est généralement indépendante de l'algorithme ; elle est le plus souvent égale à 32, 64 ou 128.

La connaissance de ces courbes est évidemment un élément essentiel pour le choix de la taille de pages. Une partie des résultats que nous avons obtenus sont présentés sur la figure 13. Comme $F(512)$ est nettement plus élevé que $F(256)$, la page de 256 mots semble une solution préférable à celle qui a été adoptée en fait pour le calculateur CII 10070.

Notons que dans cette machine, il y a autant de registres topographiques que de pages physiques dans la configuration maximale. Pour diviser la taille des pages par 2, il faut soit renoncer à cet avantage, soit accepter une augmentation du coût de la topographie. Par ailleurs, le choix de la taille de page doit prendre en compte les caractéristiques des deux niveaux de la hiérarchie de mémoire, ce qui est très complexe.

La figure 13 montre également que le produit $TPAGE \times F(TPAGE)$, c'est-à-dire le taux de transfert de mots, est une fonction croissante de $TPAGE$.

Nous avons également étudié l'efficacité de l'algorithme A_0 : dans une mémoire physique de m pages, A_0 maintient constamment les $(m-1)$ pages du programme référencées avec la plus grande fréquence. Le taux de défaut de page de cet ARP est élevé : pour le programme C, il est égal à 0,05302 si $TPAGE = 512$, à 0,04633 si $TPAGE = 0,04633$. Ces taux sont supérieurs à ceux que nous avons noté pour FIFO et LRU. Le comportement de ce programme n'est pas conforme au modèle à références indépendantes.

6.2. INFLUENCE DE LA TAILLE DE LA MEMOIRE CENTRALE

Supposons désormais qu'une taille de page a été choisie. Soit s le nombre de pages physiques allouées à un programme donné. Nous nous sommes intéressés à la durée de vie $e(s)$, c'est-à-dire au nombre moyen de références entre deux références provoquant un défaut de page. Les courbes que nous avons obtenues sont peu régulières et il est difficile-voire impossible-de leur appliquer des approximations telles que celle de Belady et Kuehner ou celle de Chamberlin *et al.* (Voir chapitre I, paragraphe 3).

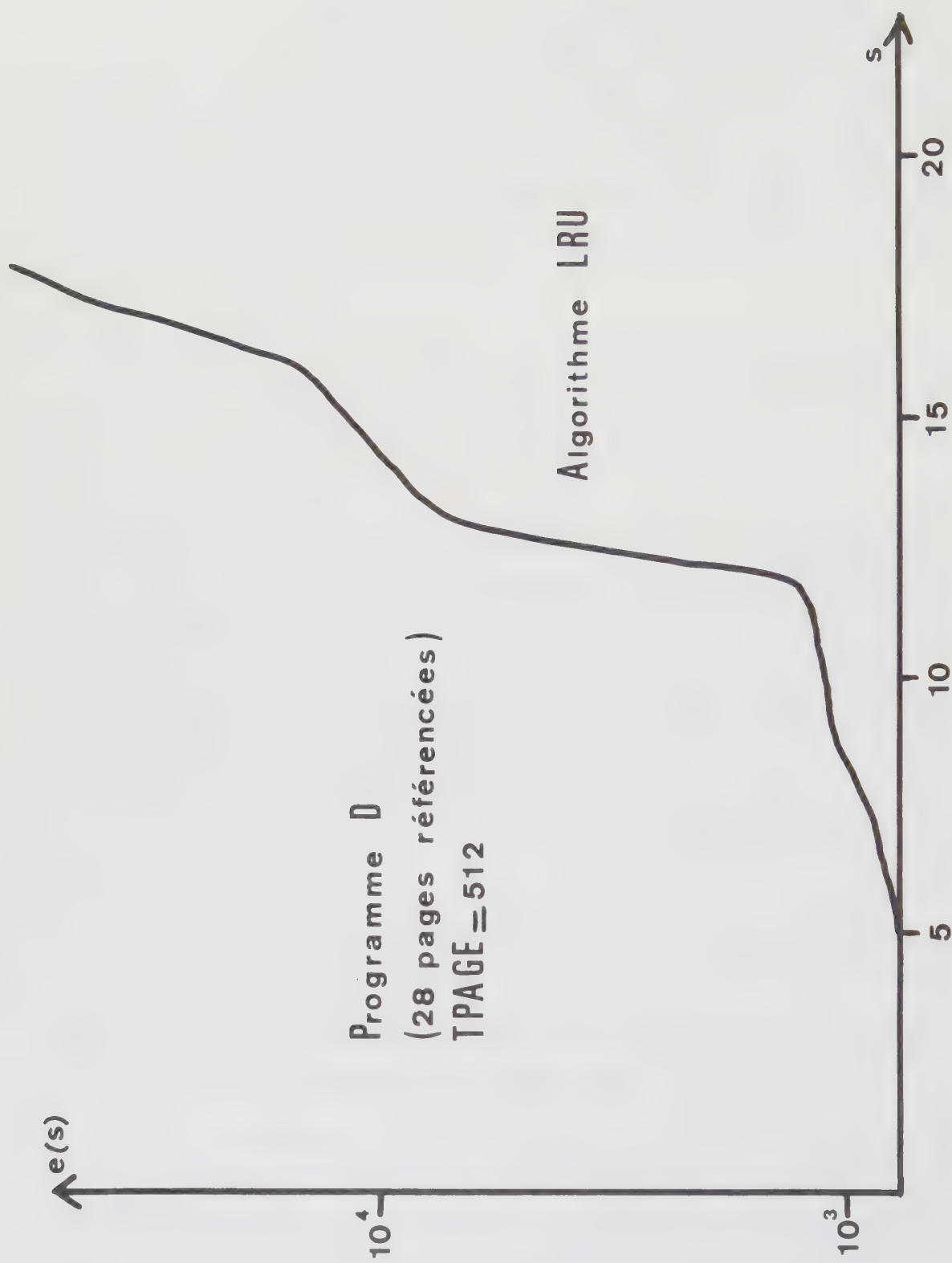
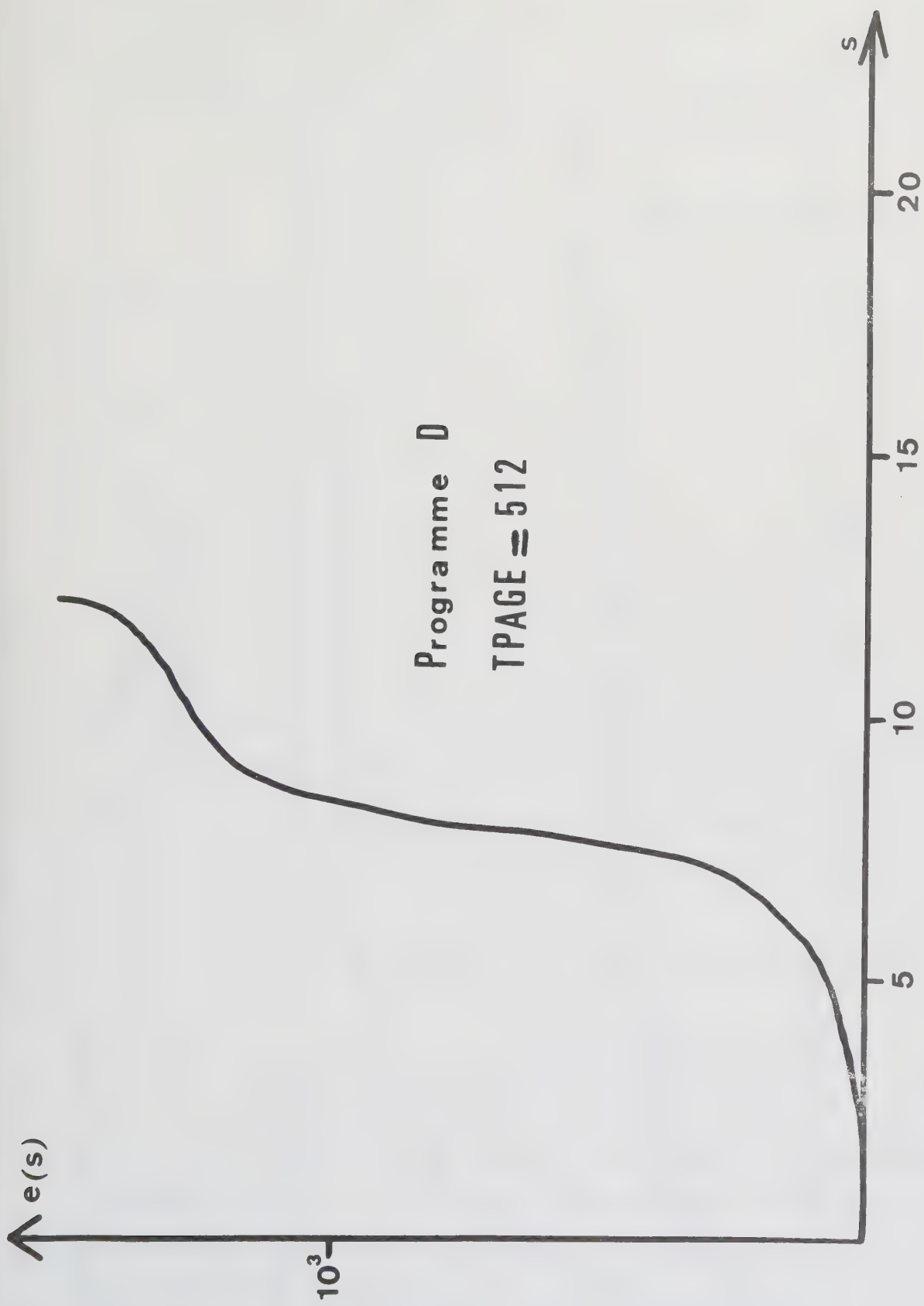


FIGURE 14



Programme D

TPAGE = 512

FIGURE 15

En tout cas, plusieurs polynômes d'approximation semblent nécessaires, selon le domaine de valeurs auquel on s'intéresse. Les figures 14 et 15 qui ne diffèrent que par l'échelle choisie sur l'axe des ordonnées, en témoignent.

7- TAILLE DE L'ENSEMBLE DE TRAVAIL

La taille de l'ensemble de travail des différents programmes de l'échantillon a été mesurée pour plusieurs valeurs de la fenêtre et plusieurs tailles de page. Les résultats ainsi obtenus se ressemblent beaucoup. La figure 16 montre la variation de la taille moyenne $\bar{w}$ (T) en fonction de la fenêtre T. Cette courbe est concave, ce que peuvent expliquer les modèles de comportement [23]. Après une croissance rapide dans le domaine des petites valeurs de T, cette fonction tend lentement vers n : la taille de la localité utile est donc très inférieure à l'espace d'adresses. Par ailleurs, la courbure devient vite très faible. La figure 16 illustre également la prédiction de la taille moyenne de l'E.T. par les modèles à références indépendantes et à liste LRU du programme B. Le premier de ces modèles se révèle complètement inadapté à la description des localités, ce qui n'est pas une surprise. En revanche, le modèle LRU conduit à une approximation assez satisfaisante de la courbe réelle.

Ceci confirme une étude antérieure de Denning *et al* [11] [12].

Nous, nous sommes également intéressés à la distribution de la variable aléatoire $w(T)$, supposée stationnaire. L'approximation gaussienne (voir chapitre 4) est assez proche de la réalité dans la majorité des cas (figure 17.a). Il n'en reste pas moins qu'elle est fréquemment erronée (figure 17.b). Les modèles de comportement considérés jusqu'ici ne sont donc pas très corrects de ce point de vue. Leur défaut est qu'ils ne permettent pas de décrire les corrélations entre l'entrée dans la localité courante de certaines pages.

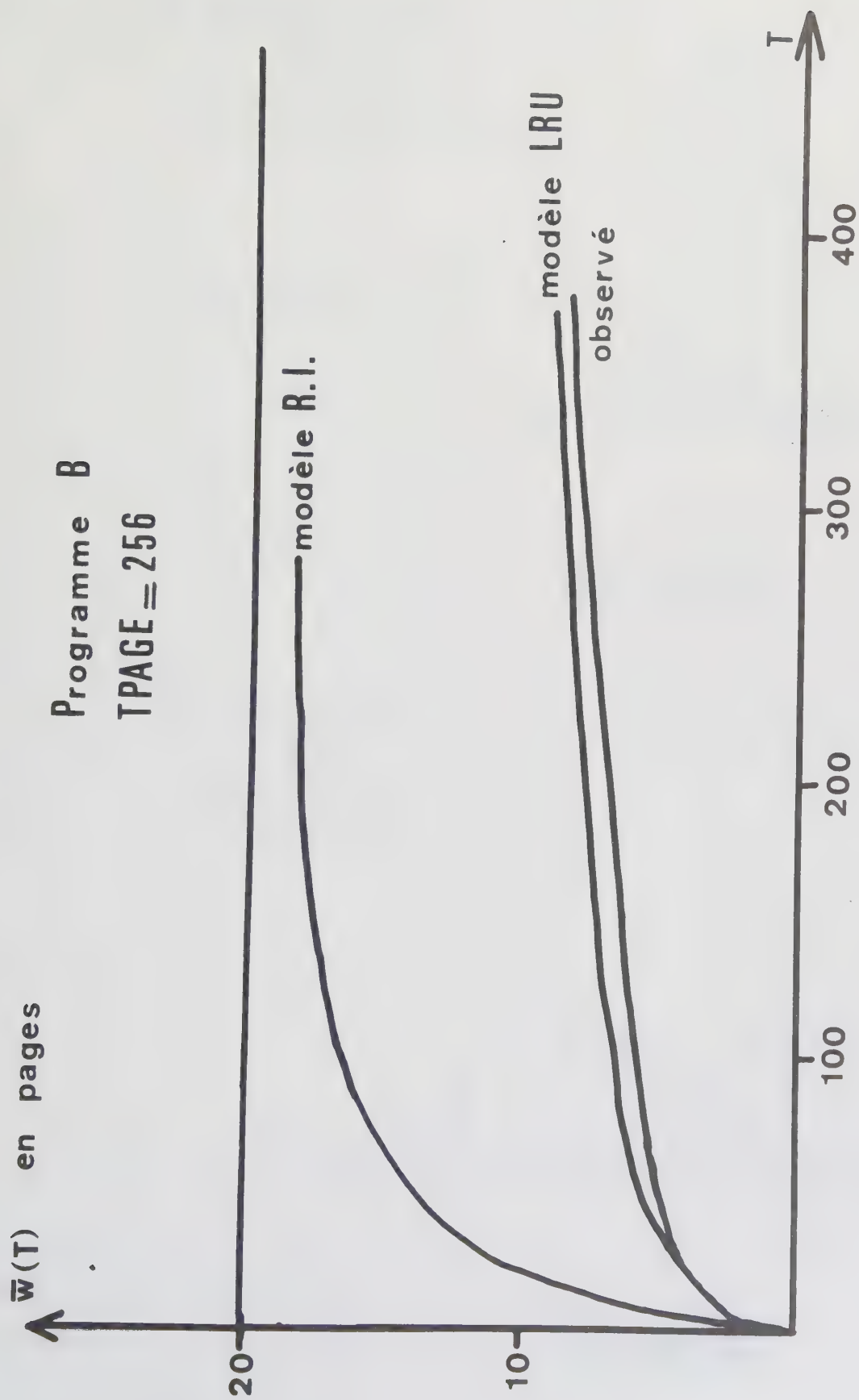
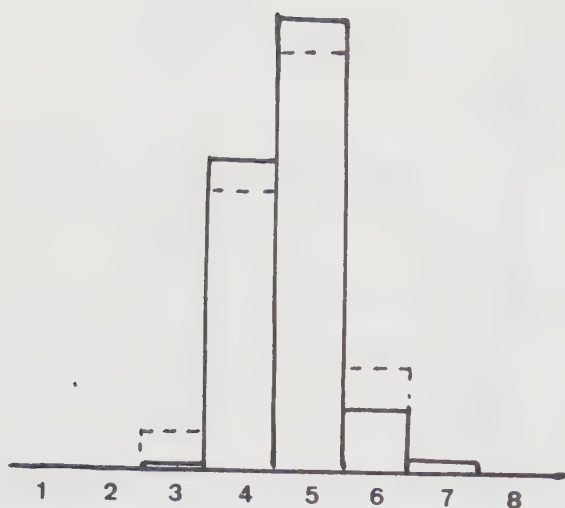


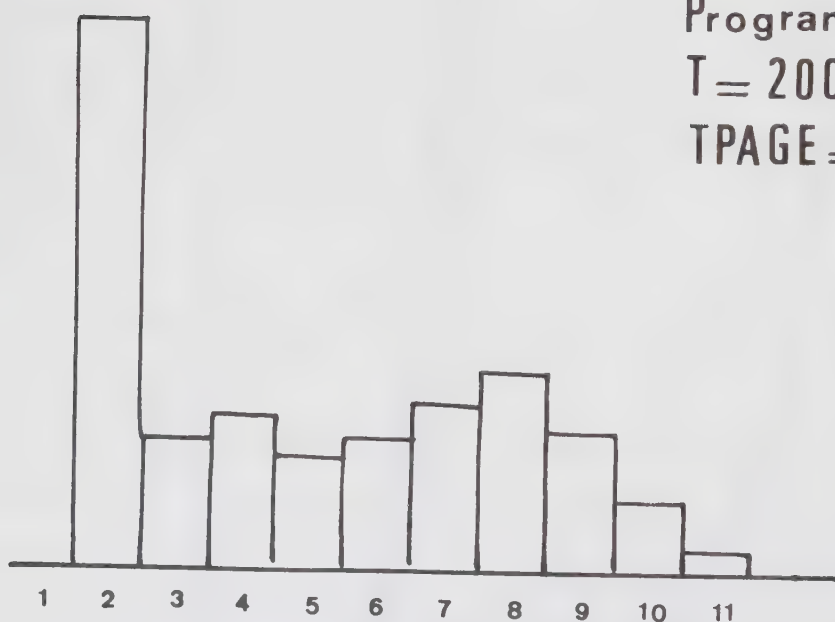
FIGURE 16



Programme D
 $T = 200$
 $TPAGE = 512$

--- APPROXIMATION
 GAUSSIENNE

FIGURE 17 a



Programme C
 $T = 200$
 $TPAGE = 512$

FIGURE 17 b

Un modèle tenant compte de cet aspect du comportement des programmes est présenté au chapitre 6 (paragraphe 6).

REFERENCES

- [1] CII 10070 Computer - Description manual - C 900 950
- [2] BAYLIS, M.H.J., FLETCHER, D.G., HOWARTH : Paging studies made on the ICT Atlas Computer - Proc. IFIP Congress 1968, 831-837 (1968)
- [3] SAYRE, D : Is automatic folding of programs efficient enough to displace manual ? - Comm. ACM 13, 12, 656-660 (1969)
- [4] BELADY, L.A., KUEHNER, C.J : Dynamic space sharing in computer systems - Comm. ACM 12, 5, 282-288 (1969)
- [5] CHU, W.W., OLIVER, N., OPDERBECK, H. : Measurement data on the working set replacement algorithm and their applications - Proc. of the Symposium on Computer - Communications and Teletraffic, Polytechnic Institute of Brooklyn, 113-124 (1972)
- [6] BRAWN, B.S., GUSTAVSON, F.G. : Program behaviour in a paging environment - Proc. AFIPS Fall JCC 1968, 1019-1031 (1968)
- [7] BARD, Y. : Characterization of program paging in a time-sharing environment - IBM J. Res. Develop. 17, 5, 387-393 (1973)
- [8] HATFIELD, D.J. : Experiments on page size, program access patterns and virtual memory performance - IBM J. Res. Develop. 16, 1, 58-66 (1972)
- [9] RODRIGUEZ - ROSELL, J. : The working set behaviour of some programs - Report NA 72.51, Royal Institute of Technology, Stockholm (1972)

- [10] RODRIGUEZ - ROSELL, J. : Experimental data on how program behaviour effects the choice of scheduler parameters - Proc. 3rd ACM Symposium on Operating Systems Principles, 156-163 (1971)

- [11] DENNING, P.J., SAVAGE, J.E., SPIRN, J.R. : Some thoughts about locality in program behaviour - Proc. of the Symposium on Computer - Communications and Teletraffic, Polytechnic Institute of Brooklyn, 101-112 (1972)

- [12] SPIRN, J.R., DENNING, P.J. : Experiments with program locality - Proc. of AFIPS fall JCC 1972, 611-621 (1972)

- [13] BRUNDAGE, R.E., BATSON, A.P. : The use of associative memory in symbolically - segmented virtual memory systems - University of Virginia, Charlottesville.

- [14] WILLIAMS, J.G. : Experiments in page activity determination - Proc. of AFIPS Spring JCC 1972, 739-747 (1972)

- [15] COFFMAN, E.G., VARIAN, L.C. : Further experimental data on the behaviour of programs in a paging environment - Comm. ACM 11, 7, 471-474 (1968)

- [16] JOSEPH, M. - An Analysis of paging and program behaviour - Comput. J. 13, 1 (1970)

- [17] FINE, G.H., JACKSON, C.W., McISAAC, P.V. : Dynamic Program behaviour under paging - Proc 21st National Conference of ACM, 223-228 (1966)

- [18] LEWIS, P.A.W., SHEDLER, G.S. : Empirically derived micromodels for sequences of page exceptions - IBM J. Res. Develop. 17, 2, 86-100 (1973)

- [19] INGELS, P., JAMES, M., LOCU, H., RAYNAL, M. : Simulateur du calculateur CII 10070 - Rapport de fin d'études, INSA de Rennes (1973)

- [20] KNUTH, D.E. : An empirical study of Fortran programs - Report CS-186, Stanford University (1970)

- [21] COFFMAN, E.G., McKELLAR, A.C. : Organizing matrices and matrix operations for paged memory systems - Comm. ACM 12, 3 (1969)

- [22] BELADY, L.A. : A study of replacement algorithms for a virtual storage computer - IBM System J. 5, 2, (1966)

- [23] DENNING, P.J., SCHWARTZ, S.C. : Properties of the working set model - Comm. ACM 15, 191-198 (1972)

ANNEXE

Regroupement par fonctions des instructions de base
du calculateur CII 10070

Chargement-Rangement : LI LB LH LW LD LS LM XW STH STB STW
STD STS STM DL DST PSW PLW PSM PLM

Branchement : EXU BCS BCR BIR BAL

Comparaison : CI CB CH CW CD CS CLR CLM DC

Arithmétique Entière : LCH LAH LCW LAW LCD LAD AI AH AW
AD SH SW SD MI MH MW DH DW AWM MTB MTH MTW

Arithmétique Réelle : FAS FAL FSS FSL FMS FML FDS FDL

Arithmétique en Base 10 : DA DS DM DD PACK UNPK

Logique : OR EOR AND

Appel au Moniteur : CAL1 CAL2 CAL3 CAL4

Décalage : S SF DSA

Traitement de chaînes de caractères : EBS MBS CBS TBS TTBS

Autres : LCFI LCF STCF ANLZ INT CVA CVC MCF

(Les instructions figurant dans ce classement sont celles qui peuvent être exécutées en mode esclave).

CHAPITRE SIX

LE MODELE A RESEAU DE RETARDS DE COMPORTEMENT DES PROGRAMMES

Le modèle à références indépendantes néglige les propriétés locales du comportement des programmes : ceci implique qu'il conduit à surestimer la taille des ensembles de travail ou le taux de défauts de page dans la gestion des mémoires paginées. En revanche ces propriétés locales sont les seules prises en compte par le modèle à liste L R U. En outre dans ce dernier toutes les pages ont des caractéristiques identiques (fréquence des références, distribution de l'intervalle entre deux références successives à une page donnée...) ; l'identité de chaque page est oubliée dans la pratique.

Nous allons présenter dans ce chapitre un nouveau modèle de comportement [1] qui allie la diversification des pages à la description du comportement local. Une extension de ce modèle (paragraphe 6) prend en compte un aspect ignoré par la plupart des modèles présentés au chapitre 1 : l'existence de corrélations entre l'apparition de certaines pages dans l'ensemble de travail. Comme nous l'avons déjà noté, les causes de ces corrélations sont nombreuses : boucles s'étendant sur plusieurs pages, procédure faisant appel à d'autres procédures...

D'autres extensions du modèle peuvent être facilement imaginées. Pour cette raison l'utilisation des réseaux de retards peut être considérée comme une méthode de description du comportement des programmes. Elle se prête particulièrement bien aux études en simulation. L'analyse de cette classe de modèles est tributaire des progrès de la théorie des réseaux de files d'attente.

1 - DESCRIPTION DU MODELE A RESEAU DE RETARDS

1.1. CARACTERISTIQUES GENERALES

L'espace d'adresses du programme est un ensemble N de n pages :

$$N = \{ 1, 2, \dots, n \}$$

Ce modèle est un modèle à temps continu. Toute exécution génère une suite croissante de réels positifs $\left(t_i \right)_{i \in \mathbb{N}}$ et une suite de pages $\left(p_i \right)_{i \in \mathbb{N}}$ telles que :

- $t_0 = 0$
- pour tout i , p_i est la page utilisée dans l'intervalle de temps $[t_i, t_{i+1} [$
- la suite $\left(t_i \right)$ n'est pas bornée ce qui revient à supposer que la durée de l'exécution est infinie.

Puisque nous nous intéressons au comportement intrinsèque du programme, le demi-axe $[0, \infty[$ est interprété comme le "temps du programme". En raison de l'importance du nombre d'accès à la mémoire qui s'exécutent dans tout intervalle de temps considéré dans la gestion des ressources, cette hypothèse de continuité est raisonnable. En outre le temps qui sépare deux références successives n'est pas constant en raison de la différence de nature des instructions exécutées.

En ce qui concerne une page donnée i , elle est alternativement active (c'est-à-dire que les adresses référencées appartiennent à i) ou en suspens (voir fig. 1). La longueur d'une période d'activité sera notée Z_i . Le temps de suspension, désigné par Q_i , sera modélisé par une variable aléatoire dont la densité a une transformée de Laplace rationnelle : cette densité est donc

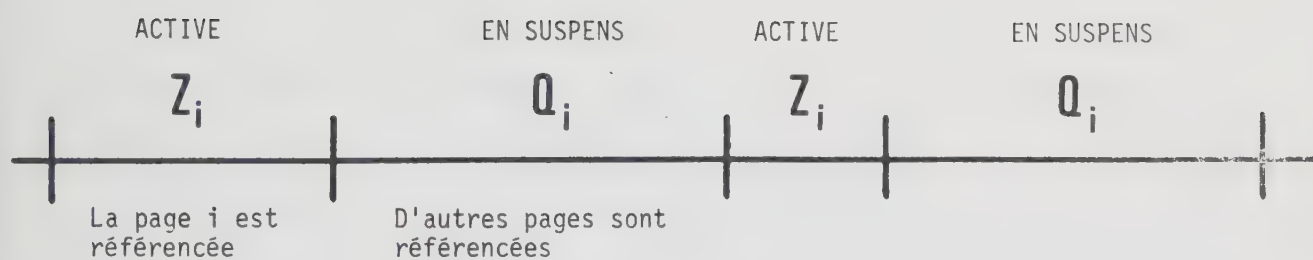


FIGURE 1

une combinaison linéaire de produits de convolution de densités exponentielles. Deux raisons justifient le choix de cette hypothèse sur Q_j . D'une part la classe de variables aléatoires considérée fournit de bonnes approximations de la plupart des distributions. D'autre part ces variables aléatoires se prêtent à une interprétation "par étapes", ainsi que Cox l'a fait remarquer [2]. Soit, en effet, une variable aléatoire Q dont la densité a une transformée de Laplace rationnelle. Il existe un réseau d'étapes, analogue à celui de la figure 2, tel que Q soit le temps nécessaire à un processus stochastique pour passer de l'état 1 à l'état absorbant. La durée du séjour du processus dans l'état j ($1 \leq j \leq r$) a une distribution exponentielle de moyenne $1/\mu_j$; lorsqu'il quitte l'état j , le processus entre dans l'état $j + 1$, avec probabilité a_j , ou dans l'état absorbant, avec probabilité $b_j = 1 - a_j$; les durées de séjour et les transitions sont stochastiquement indépendantes.

En raison du principe de localité, on peut s'attendre à ce que le séjour dans les premiers états soit moins long que dans les derniers : les taux $\mu_1, \mu_2, \dots$ seraient plus élevés que $\mu_r, \mu_{r-1}, \dots$. Alors que l'on peut considérer que les premiers états modèlent le comportement d'une page lorsqu'elle se trouve dans la localité active, on diversifie l'importance relative des pages en choisissant convenablement les taux et les probabilités de transition d'indice élevé.

Au paragraphe 1.2. les relations entre les différentes pages sont définies à l'aide d'un réseau de retards. Ce modèle sera utilisé dans les paragraphes suivants pour analyser la taille de l'ensemble de travail et pour calculer le taux de défauts de page résultant de l'emploi de l'algorithme de remplacement L R U.

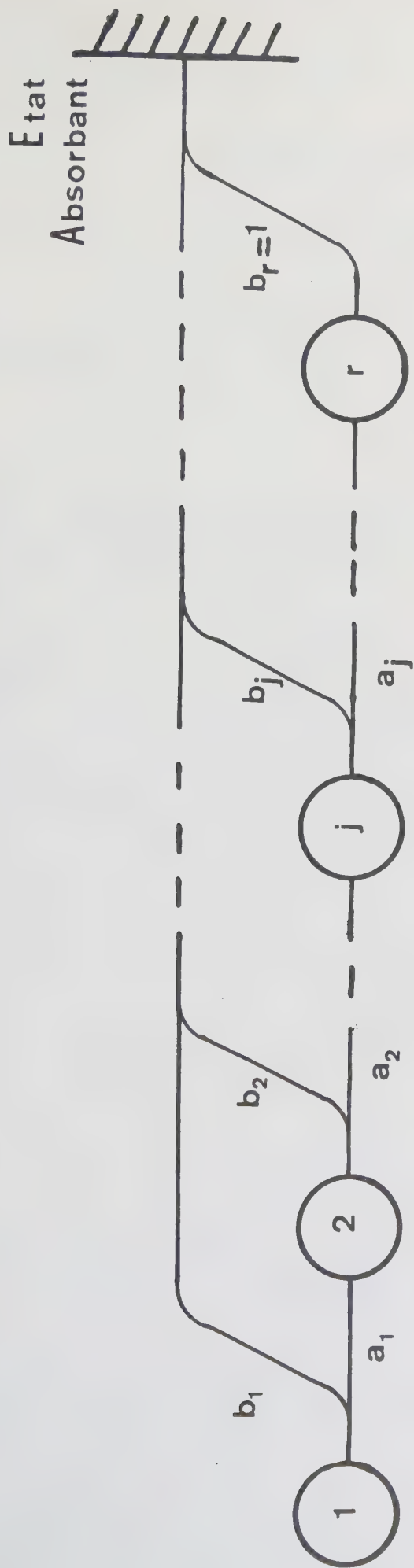


FIGURE 2

1.2. DEFINITION DU MODELE A RESEAU DE RETARDS

Le modèle à réseau de retards d'un programme est déterminé par la connaissance des éléments suivants :

- i) l'ensemble $N = \{ 1, 2, \dots, n \}$ de ses pages
- ii) la famille finie $\left\{ r_i \right\}_{i \in N}$ d'entiers strictement positifs, où r_i est le nombre d'états dans la décomposition de Cox de Q_i .

Soit S^* l'ensemble des couples d'entiers (i, j) tels que $1 \leq i \leq n$ et $1 \leq j \leq r_i$; $S^* \cup \{0\}$ est l'ensemble des états de page. Nous désignerons par S l'ensemble des n -uples ordonnés

$$\left(j_1, j_2, \dots, j_i, \dots, j_n \right) \text{ tels que}$$

- 1) pour tout i , j_i est un entier et $0 \leq j_i \leq r_i$
- 2) il y a un entier k unique ($1 \leq k \leq n$) tel que $j_k = 0$

L'ensemble S apparaîtra comme l'ensemble des états du programme.

- iii) la famille de réels strictement positifs $\left\{ \mu_{ij} \right\}_{(i,j) \in S^*}$, où $1/\mu_{ij}$ est la durée moyenne du séjour de la page i dans l'état (i, j)
- iv) la famille de probabilités $\left\{ a_{ij} \right\}_{(i,j) \in S^*}$; pour tout i , $a_{ir_i} = 0$. Nous noterons b_{ij} la probabilité $1 - a_{ij}$.

- v) un élément σ_0 de S appelé état initial du programme.

La i composante d'un état du programme est l'état occupé par la page i . Par exemple, le programme est dans l'état $\left(0, j_2, j_3, \dots, j_n \right)$ si et seulement si la page 1 est dans l'état 0 et la page i ($2 \leq i \leq n$) dans l'état $\left(i, j_i \right)$.

A l'instant 0 le programme est dans l'état 0. Il évolue selon les hypothèses suivantes :

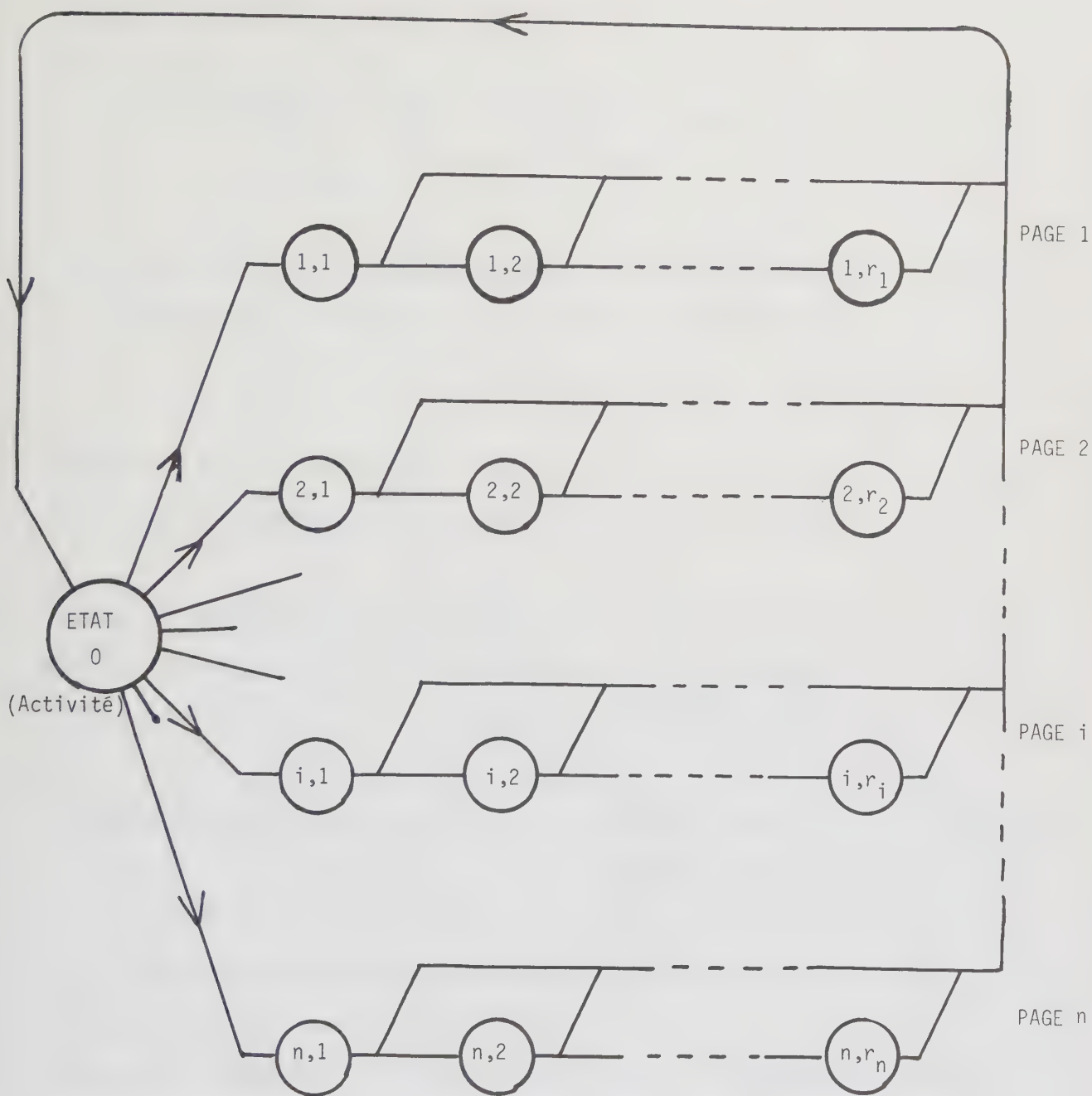


FIGURE 3

- R R 0 Seule la page i peut être dans l'état j . Nous dirons donc "l'état j de la page i " pour (i,j) .
- R R 1 La durée du séjour de la page i dans l'état j est une variable aléatoire D_{ij} à distribution exponentielle de paramètre λ_{ij} .
- R R 2 En quittant l'état j , la page i entre dans l'état $j + 1$ (avec probabilité a_{ij}) ou dans l'état 0 (avec probabilité b_{ij}).
- R R 3 Si la page i est dans l'état 0, elle y reste jusqu'à ce qu'une autre page entre dans cet état : dès que cet événement survient, la page i passe dans l'état $(i,1)$.
- R R 4 Les durées de séjour D_{ij} et les transitions sont stochastiquement indépendantes.

Il est clair que R R 0 est une conséquence des autres hypothèses (de R R 3 en particulier). Comme l'état initial du programme appartient à S, il résulte de l'hypothèse R R 3 qu'à tout instant il y a une page - et une seule - dans l'état 0 : cette page est la page référencée à l'instant considéré. Une page est dans un état $j \geq 1$ si et seulement si elle est en suspens.

1.3. COMPARAISON AVEC LE MODELE A REFERENCES INDEPENDANTES ET LE MODELE A LISTE L R U

Dans le modèle à références indépendantes, la période de suspension de la page i a une durée distribuée géométriquement avec paramètre $1 - \beta_i$. Ce

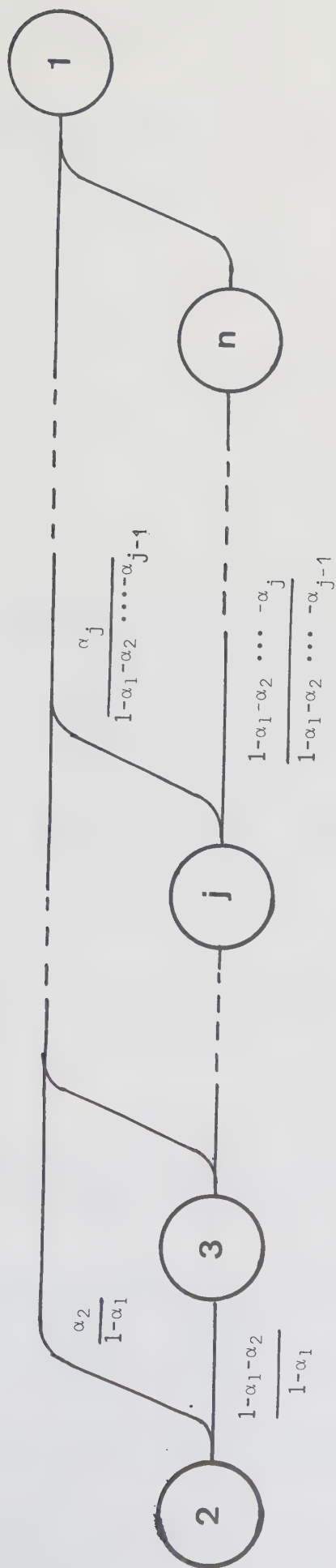


FIGURE 4

modèle est presque l'analogue discret d'un réseau de retard tel que $r_i = 1$ pour tout i .

Dans le modèle à liste L R U, la page référencée est à la distance 1 dans la liste. Soit Q la durée entre l'instant où une page donnée quitte la distance 1 et l'instant où elle rentre dans cet état pour la première fois. Une décomposition de Q analogue à celle de Cox est montrée sur la figure 4 : les numéros d'étape sont les distances et le temps passé dans l'état j ($2 \leq j \leq n$) a une distribution géométrique de rapport $\alpha_1 + \alpha_2 + \dots + \alpha_j - 1$. Une différence essentielle entre le modèle à liste L R U et celui à réseau de retards réside dans les relations de dépendance linéaire entre les inverses des durées moyennes d'étape et les probabilités de transition. Ces relations restreignent effectivement la classe des distributions possibles pour Q dans le modèle à liste L R U : la distribution de fonction génératrice $\left(\frac{1}{1 - \alpha z} \right)^n$ n'appartient pas à cette classe alors que son analogue continu, une distribution d'Erlang, peut figurer dans le modèle à réseau de retards. En outre dans le modèle à liste L R U, la distribution de Q est la même pour toutes les pages. Toutefois le modèle à liste L R U ne peut être considéré comme un cas particulier du modèle à réseaux de retards à cause de la manière dont les références aux pages sont corrélées.

2 - RESULTATS FONDAMENTAUX SUR LE MODELE A RESEAUX DE RETARDS

Pour tout état de page (i, j) définissons le nombre A_{ij} par :

$$(1) \quad A_{ij} = \begin{cases} 1 & \text{si } j = 1 \\ a_{i1} \times a_{i2} \times \dots \times a_{i,j-1} & \text{si } j > 1 \end{cases}$$

Les trois formules ci-dessous, qu'il est facile de vérifier nous seront très utiles :

$$(2) \quad \sum_{i=1}^{r_i} A_{ij} b_{ij} = 1 \quad \text{pour } 1 \leq i \leq n$$

$$(3) \quad A_{i, j-1} - A_{ij} = A_{i, j-1} b_{i, j-1} \quad \text{pour } 1 \leq i \leq n \text{ et } 2 \leq j \leq r_i$$

$$(4) \quad \bar{Q}_i = \sum_{j=1}^{r_i} \frac{A_{ij}}{\mu_{ij}} \quad \text{pour } 1 \leq i \leq n, \bar{Q}_i \text{ désignant l'espérance mathématique de } Q_i$$

2.1. DISTRIBUTION A L'EQUILIBRE DE L'ETAT DU PROGRAMME

Soit $\sigma \in S$ un état de programme. D'après le théorème ergodique, la probabilité de trouver le programme dans l'état σ à l'instant t a une limite strictement positive $\pi(\sigma)$ lorsque t tend vers l'infini ; $\pi(\sigma)$ ne dépend pas de l'état initial σ_0 . Dans ce paragraphe nous allons étudier la distribution d'équilibre $\pi(\sigma)$.

Notons $c(\sigma)$ le rang de la composante nulle dans σ , c'est-à-dire le numéro de la page active.

Proposition 1

Pour tout n-uple $\sigma = (j_1, j_2, \dots, j_n)$ de S , la probabilité d'équilibre est

$$(5) \quad \pi(\sigma) = G^{(n)} \prod_{\substack{i=1 \\ i \neq c(\sigma)}}^n \frac{A_{i, j_i}}{\mu_{i, j_i}}$$

où $G^{(n)}$ désigne la constante de normalisation

$$(6) \quad G^{(n)} = \left(\prod_{i=1}^{j_1} \overline{Q}_i \times \sum_{i=1}^{j_1} \frac{1}{\overline{Q}_i} \right)^{-1}$$

Démonstration : Pour tous $1 \leq i$, $l \leq n$, la relation ci-dessous définit une fonction sur l'ensemble des états de programme dont la i è composante a la valeur 1 :

$$\Gamma_{il}(\sigma) = \left(j_1, \dots, j_{i-1}, 0, j_{i+1}, \dots, j_{c(\sigma)-1}, l, j_{c(\sigma)+1}, \dots, j_n \right)$$

si et seulement si

$$\sigma = \left(j_1, \dots, j_{i-1}, 1, j_{i+1}, \dots, j_{c(\sigma)-1}, 0, j_{c(\sigma)+1}, \dots, j_n \right)$$

De même pour tout $1 \leq l \leq n$ et tout état de programme

$$\sigma = \left(j_1, \dots, j_{l-1}, j_l, j_{l+1}, \dots, j_n \right)$$

tel que $j_l < r_l$, nous posons

$$\Delta_l(\sigma) = \left(j_1, \dots, j_{l-1}, j_l + 1, j_{l+1}, \dots, j_n \right)$$

Nous voyons que lorsque le programme entre dans l'état σ il provient d'un état de la forme $\Gamma_{il}(\sigma)$ ou $\Delta_l(\sigma)$. Pour tout $\sigma \in S$, l'équation d'équilibre du programme est donc :

$$(7) \quad \pi(\sigma) \sum_{\substack{i=1 \\ i \neq c(\sigma)}}^n \mu_{i,j_i} = \sum_{\substack{i \text{ tel que } l=1 \\ j_i=1}} \sum_{l=1}^{r_i} \pi \left(\Gamma_{il}(\sigma) \right) \mu_{c(\sigma),l} b_{c(\sigma),l} +$$

$$\sum_{\substack{i \text{ tel que} \\ j_i \neq 0,1}} \pi \left(\Delta_i (\sigma) \right) \mu_{i,j_i-1} a_{i,j_i-1}$$

Pour montrer que (5) est une solution de ces équations remarquons que

$$(8) \quad \pi \left(\Gamma_{i\ell} (\sigma) \right) = \pi(\sigma) \times \frac{A_{c(\sigma),\ell}}{\mu_{c(\sigma),\ell}} \times \mu_{i1}$$

$$(9) \quad \pi \left(\Delta_i (\sigma) \right) = \pi(\sigma) \times \frac{\mu_{ij_i}}{a_{i,j_i-1}} \times \frac{1}{\mu_{i,j_i-1}}$$

Il est donc suffisant de vérifier que

$$\begin{aligned} \pi(\sigma) \sum_{\substack{i=1 \\ i \neq c(\sigma)}}^n \mu_{i,j_i} &= \pi(\sigma) \sum_{\substack{i \text{ tel que} \\ j_i = 1}} \sum_{\ell=1}^{r_c(\sigma)} \frac{A_{c(\sigma),\ell}}{\mu_{c(\sigma),\ell}} \mu_{c(\sigma),\ell} b_{c(\sigma),\ell} \mu_{i1} + \\ &\pi(\sigma) \sum_{\substack{i \text{ tel que} \\ j_i \neq 0,1}} \frac{\mu_{ij_i}}{a_{i,j_i-1}} \frac{1}{\mu_{i,j_i-1}} \mu_{i,j_i-1} a_{i,j_i-1} \end{aligned}$$

Ce qui équivaut à

$$(10) \quad \sum_{\substack{i=1 \\ i \neq c(\sigma)}}^n \mu_{i,j_i} = \sum_{\substack{i \text{ tel que} \\ j_i = 1}} \mu_{i1} \sum_{\ell=1}^{r_c(\sigma)} A_{c(\sigma),\ell} b_{c(\sigma),\ell} + \sum_{\substack{i \text{ tel que} \\ j_i \neq 0,1}} \mu_{ij_i}$$

Cette dernière égalité résulte de l'égalité (2). Pour achever la démonstration, il nous reste à calculer la constante de normalisation. Puisque nous devons avoir $\sum_{\sigma} \pi(\sigma) = 1$, il faut que

$$\left(G^{(n)}\right)^{-1} = \sum_{\sigma \in S} \prod_{\substack{i=1 \\ i \neq c(\sigma)}}^n \frac{A_{ij_i}}{\mu_{ij_i}} = \sum_{k=1}^n \sum_{\sigma \text{ tel que } c(\sigma)=k} \prod_{\substack{i=1 \\ i \neq k}}^n \frac{A_{ij_i}}{\mu_{ij_i}} = \sum_{k=1}^n \prod_{\substack{i=1 \\ i \neq k}}^n \sum_{j=1}^{r_i} \frac{A_{ij}}{\mu_{ij}}$$

L'égalité (4) nous permet de simplifier cette expression :

$$\left(G^{(n)}\right)^{-1} = \sum_{k=1}^n \prod_{\substack{i=1 \\ i \neq k}}^n \bar{Q}_i = \prod_{i=1}^n \bar{Q}_i \times \sum_{i=1}^n \frac{1}{\bar{Q}_i}.$$

Tout au long de ce chapitre, nous supposons que le programme est étudié suffisamment longtemps après le début de son exécution de sorte que l'influence de l'état initial a disparu et que la distribution de l'état du programme est la distribution limite.

2.2. UTILISATION DE LA PAGE i

La probabilité que la page i soit la page référencée est donnée par

$$\begin{aligned} P_i &= \sum_{\sigma \text{ tel que } c(\sigma)=i} \pi(\sigma) = G^{(n)} \times \sum_{\sigma \text{ tel que } c(\sigma)=i} \prod_{\substack{k=1 \\ k \neq i}}^n \frac{A_{ij_i}}{\mu_{ij_i}} = \\ &= G^{(n)} \prod_{\substack{\ell=1 \\ \ell \neq i}}^n \sum_{j=1}^{r_\ell} \frac{A_{\ell j}}{\mu_{\ell j}} = G^{(n)} \prod_{\substack{\ell=1 \\ \ell \neq i}}^n \bar{Q}_\ell \end{aligned}$$

Finalement nous obtenons

$$(11) \quad P_i = \frac{1/\bar{Q}_i}{\sum_{\ell=1}^n 1/\bar{Q}_\ell}$$

De cette expression, il est facile de dériver la durée moyenne $\bar{Z}_i$ d'une période pendant laquelle la page i est référencée sans interruption. Puisque

$$P_i = \frac{\bar{Z}_i}{\bar{Z}_i + \bar{Q}_i}$$

cette valeur est

$$(12) \quad \bar{Z}_i = \frac{1}{\sum_{\substack{l=1 \\ l \neq i}}^n \frac{1}{\bar{Q}_l}}$$

La distribution de Z_i peut aussi être calculée ; la densité a une expression assez compliquée mais sa transformée de Laplace est rationnelle.

2.3. LE THEOREME FONDAMENTAL

Supposons que la page i n'est pas dans l'état 0 et soit U_i le temps qui s'est écoulé depuis l'instant le plus récent où cette page a quitté l'état 0 (si la page i est dans l'état 0, nous poserons $U_i = 0$). Notons J_i la variable aléatoire dont la valeur est l'état de la page i : U_i et J_i sont évidemment corrélés. Pour toute paire $(i,j) \in S^*$ et tout $u > 0$, posons

$$(13) \quad H_{ij}(u) = \begin{cases} \Pr \left(u < D_{i1} \right) & \text{si } j = 1 \\ \Pr \left(D_{i1} + D_{i2} + \dots + D_{i,j-1} < u < D_{i1} + D_{i2} + \dots + D_{ij} \right) & \text{si } j > 1 \end{cases}$$

et

$$(14) \quad e_{ij}(u)du = \Pr \left(u < D_{i1} + D_{i2} + \dots + D_{ij} < u + du \right).$$

Autrement dit $H_{ij}(u)$ est la probabilité que la page i soit dans l'état j à l'instant $t + u$ sachant qu'elle a quitté l'état 0 à l'instant t et qu'elle n'y est pas retournée dans l'intervalle. La fonction e_{ij} est la densité de U_i conditionnée par $J_i = j$. Les fonctions H_{ij} et e_{ij} sont étroitement liées. En effet si $j > 1$

$$H_{ij}(u) = \int_0^u e_{i,j-1}(t) e^{-\mu_{ij}(u-t)} dt = \frac{1}{\mu_{ij}} \int_0^u e_{i,j-1}(t) \mu_{ij} e^{-\mu_{ij}(u-t)} dt.$$

L'intégrale de droite est la valeur au point u du produit de convolution de $e_{i,j-1}$ et d'une densité exponentielle de paramètre μ_{ij} . Nous en déduisons :

$$(15) \quad \left(\forall (i,j) \in S^* \right) \quad H_{ij} = \frac{1}{\mu_{ij}} e_{ij}$$

cette formule étant évidemment valide pour $j = 1$.

Nous allons maintenant étudier la distribution conjointe des variables aléatoires U_i . Soit σ un état du programme ; posons $c(\sigma) = k$. Nous désignerons par

$$\eta \left(\sigma ; u_1, u_2, \dots, u_{k-1}, u_{k+1}, \dots, u_n \right) du_1 du_2 \dots du_{k-1} du_{k+1} \dots du_n$$

la probabilité (au second ordre près) que les deux événements ci-dessous se produisent simultanément à un instant donné

- le programme est dans l'état σ
- $(\forall i \in N) (i \neq c(\sigma)) : u_i \leq U_i \leq u_i + du_i$

Théorème 1 Soit σ un état du programme et soient $u_1, u_2, \dots, u_{c(\sigma)-1}, u_{c(\sigma)+1}, \dots, u_n$ $(n-1)$ nombres réels. Alors :

$$(16) \quad \eta \left(\sigma ; u_1, u_2, \dots, u_n \right) = G^{(n)} \prod_{\substack{i=1 \\ i \neq c(\sigma)}}^n \frac{A_{ij_i} e_{ij_i}(u_i)}{\mu_{ij_i}}$$

$$(17) \quad = \eta(\sigma) \prod_{\substack{i=1 \\ i \neq c(\sigma)}}^n e_{ij_i}(u_i)$$

où

$$\sigma = \left(j_1, j_2, \dots, j_n \right).$$

Ce théorème est montré en appendice. On en déduit sans peine les deux corollaires ci-dessous.

Proposition 2 Sous la condition que le programme est dans l'état σ , les variables aléatoires $U_1, \dots, U_{c(\sigma)-1}, U_{c(\sigma)+1}, \dots, U_n$ sont indépendantes.

Proposition 3 Sous la condition que la page k est dans l'état 0, les variables aléatoires bidimensionnelles $\left(J_1, U_1 \right), \dots, \left(J_{k-1}, U_{k-1} \right), \left(J_{k+1}, U_{k+1} \right), \dots, \left(J_n, U_n \right)$ sont indépendantes. En outre pour tout $i \neq k$ et tout $1 \leq j \leq r_i$:

$$(18) \quad \text{Pr} \left(J_i = j ; u \leq U_i \leq u + du \mid \text{la page } k \text{ est dans l'état } 0 \right) = \frac{A_{ij} e_{ij}(u)}{\bar{Q}_i \mu_{ij}} du$$

2.4. LA DISTRIBUTION DES VARIABLES ALÉATOIRES U_i

Notons que l'entier k n'apparaît pas dans le membre de droite de l'éga-

lité (18). Par conséquent, si nous posons

$$(19) \quad \xi_i(t) = \Pr \left\{ U_i > t \mid \text{la page } i \text{ n'est pas dans l'état } 0 \right\} \\ (1 \leq i \leq n \text{ et } u > 0)$$

nous pouvons écrire

$$(20) \quad \xi_i(t) = \sum_{j=1}^{r_i} \frac{A_{ij}}{\bar{Q}_i \mu_{ij}} E_{ij}(t)$$

où nous avons noté :

$$(21) \quad E_{ij}(t) = \int_t^\infty e_{ij}(u) du$$

Les probabilités $\xi_i(t)$ seront évidemment très utiles dans le calcul de la taille de l'ensemble de travail. Afin de les calculer établissons une expression algébrique qui sera valable si la condition

$$(22) \quad (\forall i \in N) (\forall j, h) \quad \mu_{ij} = \mu_{ih} \implies j = h$$

est vérifiée. La transformée de Laplace de e_{ij} est une fonction rationnelle e_{ij}^* :

$$e_{ij}^*(s) = \frac{\mu_{i1}}{\mu_{i1}+s} \times \frac{\mu_{i2}}{\mu_{i2}+s} \times \dots \times \frac{\mu_{ij}}{\mu_{ij}+s}$$

dont la décomposition en éléments simples est

$$(23) \quad e_{ij}^*(s) = \sum_{h=1}^j C_{ij}^h \frac{\mu_{ih}}{\mu_{ih}+s}$$

le coefficient C_{ij}^h étant égal à $\prod_{\substack{k=1 \\ k \neq h}}^j \frac{\mu_{ik}}{\mu_{ik} - \mu_{ih}}$ pour tous $1 \leq i \leq n$, $1 \leq j \leq r_i$

et $1 \leq h \leq j$ (en particulier $C_{i1}^1 = 1$). Notons la relation de récurrence entre ces coefficients :

$$(24) \quad C_{ij}^h = C_{ij-1}^h \frac{\mu_{ij}}{\mu_{ij} - \mu_{ih}} \quad (j > h, j > 1)$$

De l'égalité (23), on déduit

$$e_{ij}(t) = \sum_{h=1}^j C_{ij}^h \mu_{ih} e^{-\mu_{ih} t}$$

et

$$(25) \quad E_{ij}(t) = \sum_{h=1}^j C_{ij}^h e^{-\mu_{ih} t}$$

Par conséquent

$$\xi_i(t) = \frac{1}{\bar{Q}_i} \sum_{j=1}^{r_i} \frac{A_{ij}}{\mu_{ij}} \sum_{h=1}^j C_{ij}^h e^{-\mu_{ih} t}$$

En permutant les deux sommations, nous obtenons

$$(26) \quad \xi_i(t) = \frac{1}{\bar{Q}_i} \sum_{k=1}^{r_i} e^{-\mu_{ik} t} \sum_{j=k}^{r_i} \frac{A_{ij}}{\mu_{ij}} C_{ij}^k$$

Il ne faut pas perdre de vue que cette expression n'est valide que si la condition (22) est vérifiée.

Nous allons établir une autre expression de ξ_i en comparant cette probabilité avec la densité q_i du temps de suspension Q_i . En considérant la probabilité que le saut de la page i vers l'état 0 ait lieu à partir de l'état j , on vérifie que

$$(27) \quad q_i(t) = \sum_{j=1}^{r_i} A_{ij} b_{ij} e_{ij}(t)$$

Désignons par $Q_i(\cdot)$ la fonction de répartition (complémentaire) de la variable aléatoire Q_i :

$$(28) \quad Q_i(t) = \int_t^{\infty} q_i(u) du = \sum_{j=1}^{r_i} A_{ij} b_{ij} E_{ij}(t).$$

Proposition 4 Soit t un nombre réel positif. Pour tout i , $1 \leq i \leq n$

$$(29) \quad \frac{d}{dt} \xi_i(t) = - \frac{Q_i(t)}{\bar{Q}_i}$$

Démonstration

a) Supposons d'abord que la condition (22) est vérifiée :

$$(\forall i \in N) (\forall j, h) \quad \mu_{ij} = \mu_{ih} \implies j = h$$

En considérant ξ_i comme une fonction de t , nous déduisons de l'égalité (26) :

$$(30) \quad \frac{d}{dt} \xi_i(t) = - \frac{1}{\bar{Q}_i} \sum_k e^{-\mu_{ik}t} \sum_{j=k}^{r_i} \frac{A_{ij}}{\mu_{ij}} C_{ij}^k \mu_{ik}$$

Fixons notre attention sur la sommation la plus droite et utilisons la relation de récurrence (24) :

$$\begin{aligned} \sum_{j=k}^{r_i} \frac{A_{ij}}{\mu_{ij}} C_{ij}^k \mu_{ik} &= \sum_{j=k+1}^{r_i} \frac{A_{ij}}{\mu_{ij}} \left(C_{ij}^k - C_{ij-1}^k \right) \mu_{ij} + A_{ik} C_{ik}^k = \\ &= \sum_{j=k}^{r_i-1} C_{ij}^k \left(A_{ij} - A_{i,j+1} \right) + C_{ir_i}^k A_{ir_i} \end{aligned}$$

Puisque $A_{ij} - A_{i,j+1} = A_{ij} b_{ij}$ (égalité 3) et que $b_{ir_i} = 1$, la somme ci-dessus est égale à :

$$\sum_{j=k}^{r_i} C_{ij}^k A_{ij} B_{ij}$$

Il résulte donc de l'égalité (30) :

$$\frac{d}{dt} \xi_i(t) = - \frac{1}{Q_i} \sum_{k=1}^{r_i} e^{-\mu_{ik}t} \sum_{j=k}^{r_i} C_{ij}^k A_{ij} B_{ij}$$

ou, si nous changeons l'ordre des sommations :

$$\frac{d}{dt} \xi_i(t) = - \frac{1}{Q_i} \sum_{h=1}^{r_i} A_{ih} B_{ih} \sum_{k=1}^{r_i} e^{-\mu_{ik}t} C_{ih}^k = - \frac{1}{Q_i} \sum_{h=1}^{r_i} A_{ih} B_{ih} E_{ih}(t)$$

la dernière simplification étant fournie par l'égalité (25). En comparant cette expression de $\frac{d\xi_i}{dt}$ avec la formule (28), nous établissons, dans ce cas particulier, la relation énoncée.

b) Désignons par Ω le sous-ensemble ouvert de $\mathbb{R}^{r_i}$ formé des vecteurs dont aucune composante n'est nulle. Considérées comme fonction du r_i -uplet de paramètres $(\mu_{i1}, \mu_{i2}, \dots, \mu_{ir_i})$, $\frac{d\xi_i}{dt}$ et $\frac{Q_i}{Q_i}$ sont définies et continues dans Ω .

Par suite, l'égalité

$$\frac{d}{dt} \xi_i(t) = - \frac{Q_i(t)}{Q_i} \quad (t > 0)$$

est vraie sur un fermé Ω_F de Ω . D'après la partie a) de la démonstration, Ω_F contient la partie de Ω où la condition (22) est vérifiée, c'est-à-dire un sous-ensemble dense. Donc $\Omega_F = \Omega$, ce qui achève la démonstration.

3 - L'ENSEMBLE DE TRAVAIL

Enonçons tout d'abord la définition de l'ensemble de travail de Denning dans le cadre des modèles à temps continu,

Définition Soient t et T deux nombres réels positifs tels que $T < t$. L'ensemble de travail $W(t, T)$ est l'ensemble de toutes les pages qui ont été référencées à un instant de l'intervalle $]t - T, t]$.

Définition La probabilité qu'une page n'appartenant pas à $W(t, T)$ soit référencée entre t et $t + dt$ sera notée $m(t, T) dt + o(dt)$. Le nombre $m(t, T)$ est le taux d'absence de page à l'instant t pour l'ensemble de travail de fenêtre T .

Dans ce paragraphe, nous allons calculer la distribution de la taille de l'ensemble de travail et établir une relation entre cette taille et le taux d'absence de page.

3.1. LA TAILLE DE L'ENSEMBLE DE TRAVAIL

Soient k et ℓ deux entiers tels que $1 \leq k \leq \ell \leq n$; représentons par $(\rho_1, \rho_2, \dots, \rho_\ell, \rho_{\ell+1}, \dots, \rho_n)$ une permutation sur l'ensemble $N = \{1, 2, \dots, n\}$. De la proposition 2, il résulte

$$(31) \quad \Pr \left[W(t, T) = \{\rho_1, \rho_2, \dots, \rho_\ell\} \mid \text{la page } \rho_k \text{ est dans l'état 0 à l'instant } t \right] \\ = \left(\prod_{\substack{i=1 \\ i \neq k}}^{\ell} \Pr [U_{\rho_i} < T \mid \text{la page } \rho_k \text{ est dans l'état 0 à l'instant } t] \right) \times$$

$$\left\{ \prod_{i=l+1}^n \Pr \left[U_{\rho_i} > T \mid \text{la page } \rho_k \text{ est dans l'état 0 à l'instant } t \right] \right\}$$

Grâce à l'égalité (19), nous obtenons l'expression suivante de cette probabilité conditionnelle :

$$(32) \quad \left(\prod_{\substack{i=1 \\ i \neq k}}^l [1 - \xi_{\rho_i}(T)] \right) \times \left(\prod_{i=l+1}^l \xi_{\rho_i}(T) \right)$$

Utilisons le théorème des probabilités totales :

$$(33) \quad \Pr \left[W(t, T) = \{\rho_1, \rho_2, \dots, \rho_l\} \right] = \prod_{i=1}^l \left(1 - \xi_{\rho_i}(T) \right) \times \prod_{i=l+1}^n \xi_{\rho_i}(T) \times$$

$$\sum_{k=1}^l \frac{P_{\rho_k}}{1 - \xi_{\rho_k}(T)}$$

où conformément à une définition antérieure, nous désignons par p_{ρ_k} la probabilité que la page ρ_k soit la page référencée. Posons

$$K = \prod_{i=1}^n \xi_i(T) \times \frac{1}{\sum_{i=1}^n 1/\bar{Q}_i}$$

Il résulte de l'égalité (11) :

$$\Pr \left[W(t, T) = \{\rho_1, \rho_2, \dots, \rho_l\} \right] = K \prod_{i=1}^l \frac{1 - \xi_{\rho_i}(T)}{\xi_{\rho_i}(T)} \times \sum_{k=1}^l \frac{1}{[1 - \xi_{\rho_k}(T)] \bar{Q}_{\rho_k}}$$

En considérant toutes les combinaisons de l pages, nous achevons la démonstration du théorème ci-dessous.

Théorème 2 Soit N_ℓ la famille de toutes les parties de N à ℓ éléments ($0 \leq \ell \leq n$). Soit g_ℓ l'application $N_\ell \longrightarrow \mathbb{R}$ définie par

$$(35) \quad g_\ell(\theta) = \prod_{i \in \theta} \frac{1 - \xi_i(T)}{\xi_i(T)} \times \sum_{i \in \theta} \frac{1}{\left[1 - \xi_i(T)\right] \bar{Q}_i} \quad \text{pour tout } \theta \in N_\ell$$

Alors :

$$(36) \quad \Pr \left[w(t, T) = \ell \right] = K \sum_{\theta \in N_\ell} g_\ell(\theta)$$

Puisque l'espérance de la taille d'un ensemble de travail ne dépend pas de t , nous écrirons $w(T)$ à la place de $w(t, T)$.

Proposition 5 L'espérance mathématique de la taille d'un ensemble de travail de fenêtre T est

$$(37) \quad \bar{w}(T) = n - \sum_{i=1}^n \left(1 - p_i\right) \xi_i(T).$$

Démonstration Cette formule peut être déduite du théorème 2. Néanmoins, il est plus simple de la montrer directement. Pour toute page i , définissons la fonction $\alpha_i(.,.)$ par

$$\alpha_i(t, T) = \begin{cases} 0 & \text{si } i \notin W(t, T) \\ 1 & \text{si } i \in W(t, T) \end{cases}$$

Il est clair que

$$w(t, T) = \sum_{i=1}^n \alpha_i(t, T)$$

et que, par suite

$$(38) \quad \bar{w}(t,T) = \sum_{i=1}^n \bar{\alpha}_i(t,T)$$

La valeur moyenne $\bar{\alpha}_i$ est la probabilité que la page i soit un élément de l'ensemble de travail ; donc

$$\begin{aligned} \bar{\alpha}_i(t,T) &= 1 - \Pr \left\{ i \notin W(t,T) \right\} \\ &= 1 - \Pr \left\{ i \notin W(t,T) \mid \text{la page } i \text{ n'est pas référencée à l'instant } t \right\} \\ &\quad \times \Pr \left\{ \text{la page } i \text{ n'est pas référencée à l'instant } t \right\} \\ (39) \quad &= 1 - \xi_i(T) \left(1 - p_i \right) \end{aligned}$$

En rassemblant les égalités (38) et (39), nous obtenons l'expression (37)

3.2. LE TAUX D'ABSENCE DE PAGE

Conformément à la définition de $m(t,T)$ nous pouvons écrire : $m(t,T)dt = \Pr \left\{ \text{une page qui n'appartient pas à } W(t,T) \text{ entre dans l'état 0 durant l'intervalle }]t, t+dt[\right\} + o(dt)$

Le théorème des probabilités totales a pour conséquence :

$$\begin{aligned} m(t,T) dt &= \sum_{(i,j) \in S^*} \Pr \left\{ \text{la page } i \text{ n'est pas dans l'état 0} \right\} \times \\ &\quad \Pr \left\{ \text{la page } i \text{ est dans l'état } j \mid \text{la page } i \text{ n'est pas dans l'état 0} \right\} \times \\ &\quad \Pr \left\{ U_i > T \mid \text{la page } i \text{ est dans l'état } j \right\} \times \\ &\quad \Pr \left\{ \text{la page } i \text{ entre dans l'état 0 durant }]t, t+dt[\mid \right. \\ &\quad \left. \text{la page } i \text{ est dans l'état } j \text{ et } U_i > T \right\} + o(dt) \end{aligned}$$

Donc :

$$m(t, T) dt = \sum_{i=1}^n \sum_{j=1}^{r_i} \left(1 - p_i \right) \times \frac{1}{Q_i} \times \frac{A_{ij}}{\mu_{ij}} \times E_{ij}(T) \times \mu_{ij} b_{ij}(t)$$

Nous obtenons facilement :

$$(40) \quad m(t, T) = \sum_{i=1}^n \frac{1}{Q_i} \left(1 - p_i \right) \sum_{j=1}^{r_i} A_{ij} b_{ij} E_{ij}(T) = \sum_{i=1}^n \left(1 - p_i \right) \frac{Q_i(T)}{Q_i}$$

En rapprochant cette formule des propositions 4 et 5, nous prouvons immédiatement le théorème ci-dessous :

Théorème 3 Soient t et T deux réels positifs tels que $T < t$. Alors

$$(41) \quad m(t, T) = \frac{\partial}{\partial T} \bar{w}(t, T)$$

Ce théorème est analogue à celui qui a été démontré par Denning et Schwartz sur leur modèle à temps discret (3).

4 - L'ALGORITHME L R U DE REMPLACEMENT DE PAGES

Supposons que le programme s'exécute dans une zone de mémoire centrale contenant m pages ($1 \leq m \leq n$). Chaque fois qu'un défaut de page se produit, l'algorithme L R U est utilisé pour choisir la page à remplacer.

Définition Le taux de défauts de page à l'instant t est le nombre

$f_{LRU}(m; t)$ tel que

$$f_{LRU}(m; t) dt = \Pr \left[\text{un défaut de page se produit pendant }] t, t+dt[\right] + o(dt)$$

Pour calculer ce taux, nous allons exprimer la probabilité qu'une page donnée ait un rang supérieur à m dans la liste $L R U$ (le rang de la page en train d'être référencée est 1). Soient i et k deux pages distinctes ; notons $N^*(m-1, i, k)$ l'ensemble de toutes les parties de N dont le cardinal est au moins $m-1$ et qui ne contiennent ni i ni k :

$$N^*(m-1, i, k) = \{c \in N \mid |c| \geq m-1 ; i \notin c ; k \notin c\}$$

Des égalités (18) et (20), il s'ensuit :

$$(42) \quad \Pr \left[\begin{array}{l} \text{à l'instant } t, \text{ la page } i \text{ est dans l'état } j \text{ et son rang est } > m \\ \text{la page } k \text{ est dans l'état } 0 \end{array} \right]$$

$$= \sum_{c \in N^*(m-1, i, k)} \int_0^\infty \left(\prod_{l \in c} [1 - \xi_l(u)] \right) \frac{A_{ij} e_{ij}(u)}{\bar{Q}_i \mu_{ij}} \left(\prod_{\substack{h \in c \\ h \neq k, i}} \xi_h(u) \right) du$$

Par conséquent, sous la condition que la page k soit dans l'état 0 à l'instant t , la probabilité que la page i soit dans l'état j à l'instant t et produise un défaut de page au cours de $]t, t+dt[$ est égale à

$$(43) \quad \left\{ \sum_{c \in N^*(m-1, i, k)} \int_0^\infty \left(\prod_{l \in c} [1 - \xi_l(u)] \right) \frac{A_{ij} e_{ij}(u)}{\bar{Q}_i \mu_{ij}} \left(\prod_{\substack{h \in c \\ h \neq k, i}} \xi_h(u) \right) du \right\} \mu_{ij} b_{ij} dt.$$

Désignons par $f_{ik}(t)dt$ la probabilité que la page i produise un défaut de page durant $]t, t+dt[$, étant donné que la page k se trouve dans l'état 0 à l'instant t . Sommons les expressions (43) pour une page i donnée et pour tous les j ($1 \leq j \leq r_i$) ; compte tenu de l'égalité (27), nous obtenons

$$(44) \quad f_{ik}(t) = \frac{1}{\bar{Q}_i} \sum_{c \in N^*(m-1, i, k)} \int_0^\infty \left(\prod_{l \in c} [1 - \xi_l(u)] \right) q_i(u) \left(\prod_{\substack{h \in c \\ h \neq k, i}} \xi_h(u) \right) du.$$

D'où finalement :

$$(45) \quad f_{LRU}(m;t) = \sum_{\substack{1 \leq i, k \leq n \\ i \neq k}} p_k f_{ik}(t)$$

Dans le cas général, le temps de calcul qui serait nécessaire pour un calcul effectif du taux de défauts de page, limite l'usage des formules (44) et (45) à des programmes de petite taille. Si les variables aléatoires Q_i sont identiquement distribuées, nous obtenons une expression beaucoup plus agréable :

$$f_{LRU} = -(n-1) \sum_{r=m+1}^n \binom{n-2}{r-2} \int_{u=0}^{\infty} \left(1 - \xi(u)\right)^{r-2} \xi(u)^{n-r} \xi''(u) du$$

(Nous avons utilisé la proposition 4 pour établir ce dernier résultat).

5 - EXEMPLES NUMERIQUES

Dans ce paragraphe nous illustrons les résultats établis ci-dessus en considérant un programme dont l'espace d'adresse est divisé en 16 pages. Les densités q_i des intervalles de suspension sont supposées être les suivantes :

$$\cdot \text{ pour } i = 1, 2, 3 \text{ ou } 4 : q_i(u) = 1,3947 e^{-u} - 0,4487 e^{-2u} + 0,0540 e^{-0,05 u}$$

$$\cdot \text{ pour } i = 5 \text{ ou } 6 : q_i(u) = \frac{(5)^6}{120} u^5 e^{-5u} \quad (\text{distribution d'Erlang d'ordre 6})$$

$$\cdot \text{ pour } i = 7 \text{ ou } 8 : q_i(u) = (168,89 u^2 + 48,519 u - 1,645) e^{-10u} + 0,1642 e^{-u} + 0,0304 e^{-0,01u}$$

i	r_i		j=1	j=2	j=3	j=4	j=5	j=6
1, 2	3	μ_{ij}	1	2	0,05			
3, 4		a_{ij}	0,5	0,1	0			
5, 6	6	μ_{ij}	5	5	5	5	5	5
		a_{ij}	1	1	1	1	1	0
7, 8	5	μ_{ij}	10	10	10	1	0,01	
		a_{ij}	1	0,5	0,3	0,2	0	
9, 10	2	μ_{ij}	2	2				
		a_{ij}	1	0				
11, 12	6	μ_{ij}	10	10	10	0,5	0,5	0,5
		a_{ij}	1	1	0,3	1	1	0
13, 14	3	μ_{ij}	1	1	1			
		a_{ij}	0,75	1	0			
15, 16	4	μ_{ij}	1	1	1	1		
		a_{ij}	1	1	0,75	0		

FIGURE 5

i	1,2,3,4	5,6	7,8	9,10	11,12	13,14	15,16
$\bar{Q}_i$	2,25	1,20	3,40	1,00	2,10	2,50	3,75
$\sigma(Q_i)$	6,38	0,49	24,4	0,71	3,35	1,80	1,99

Valeur moyenne et écart-type de l'intervalle de suspension

FIGURE 6

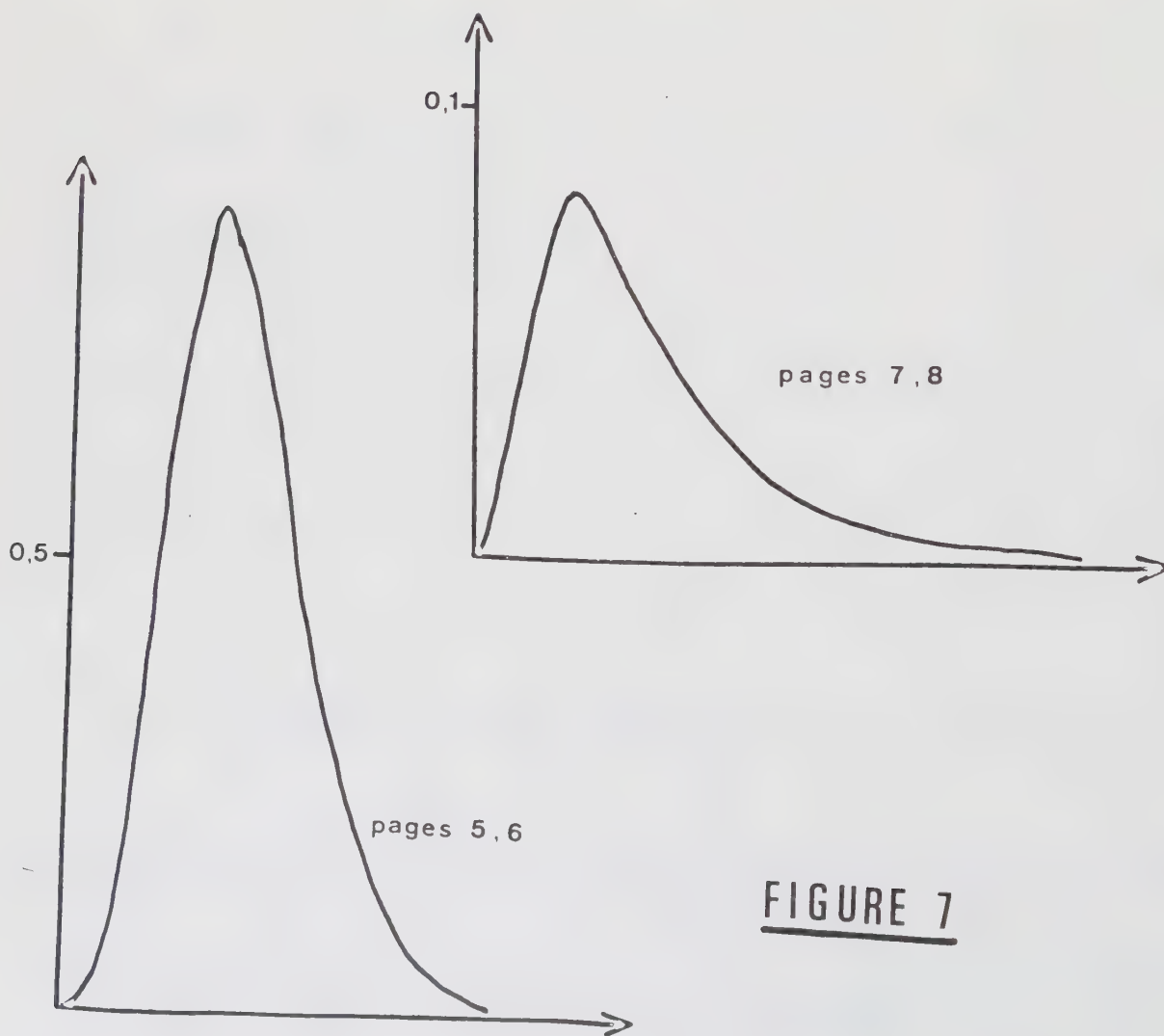
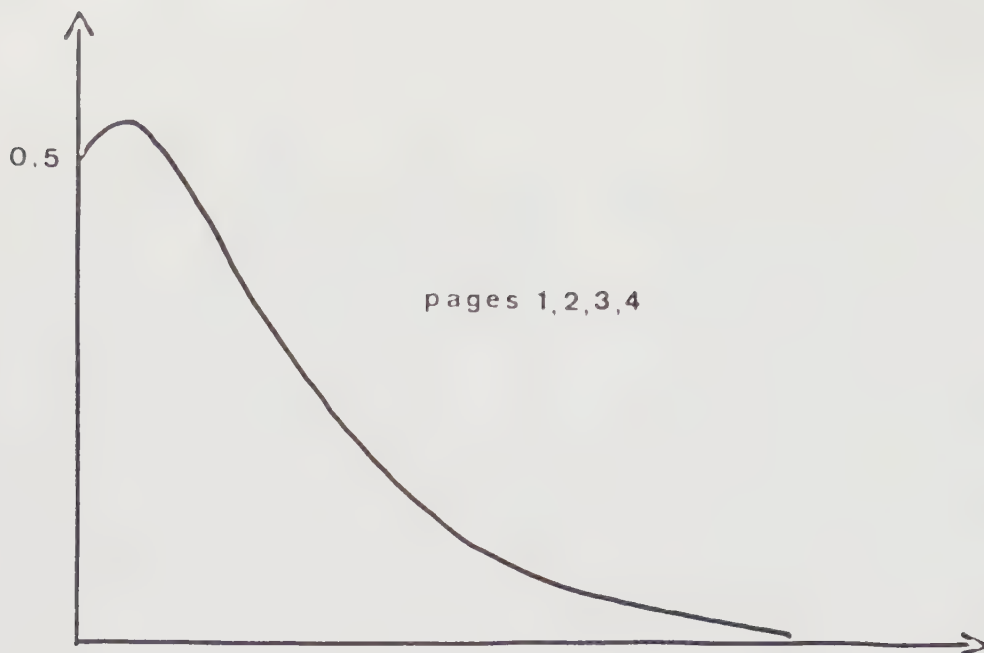
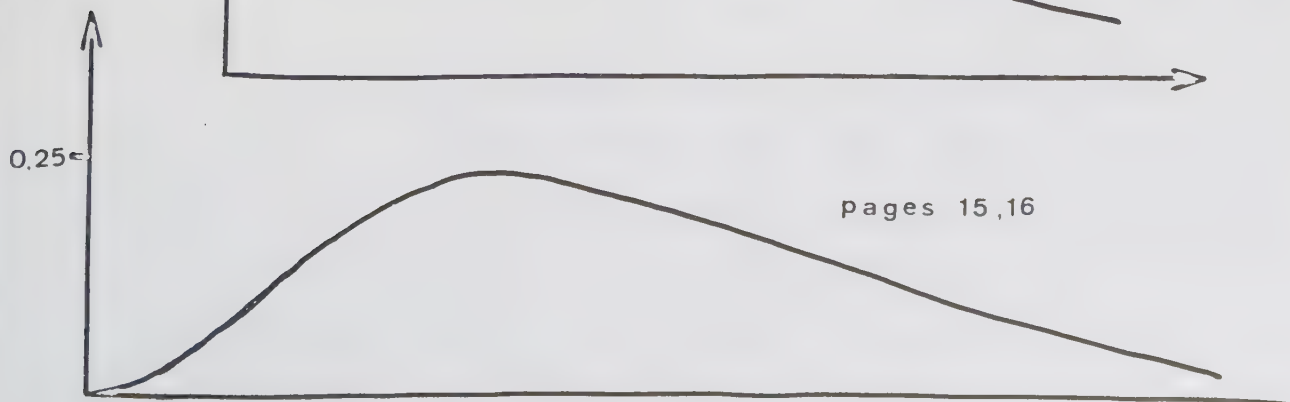
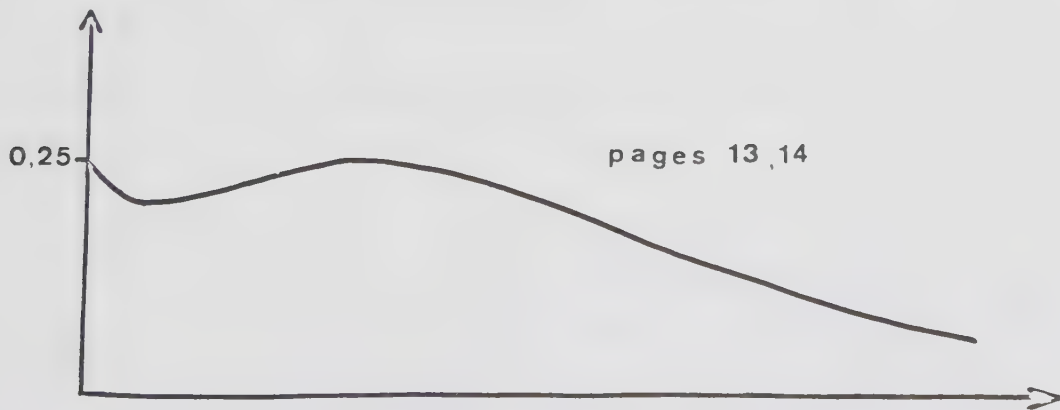
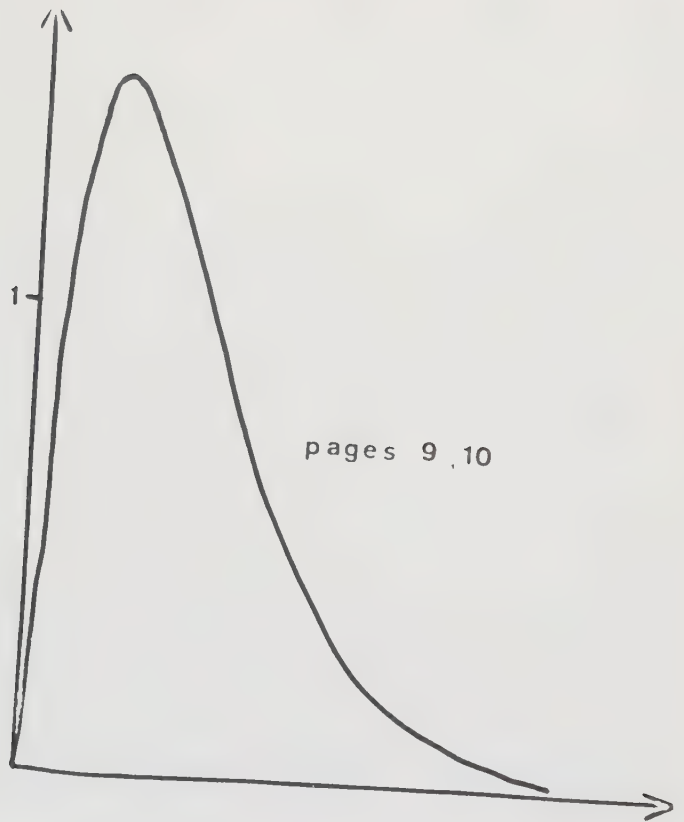
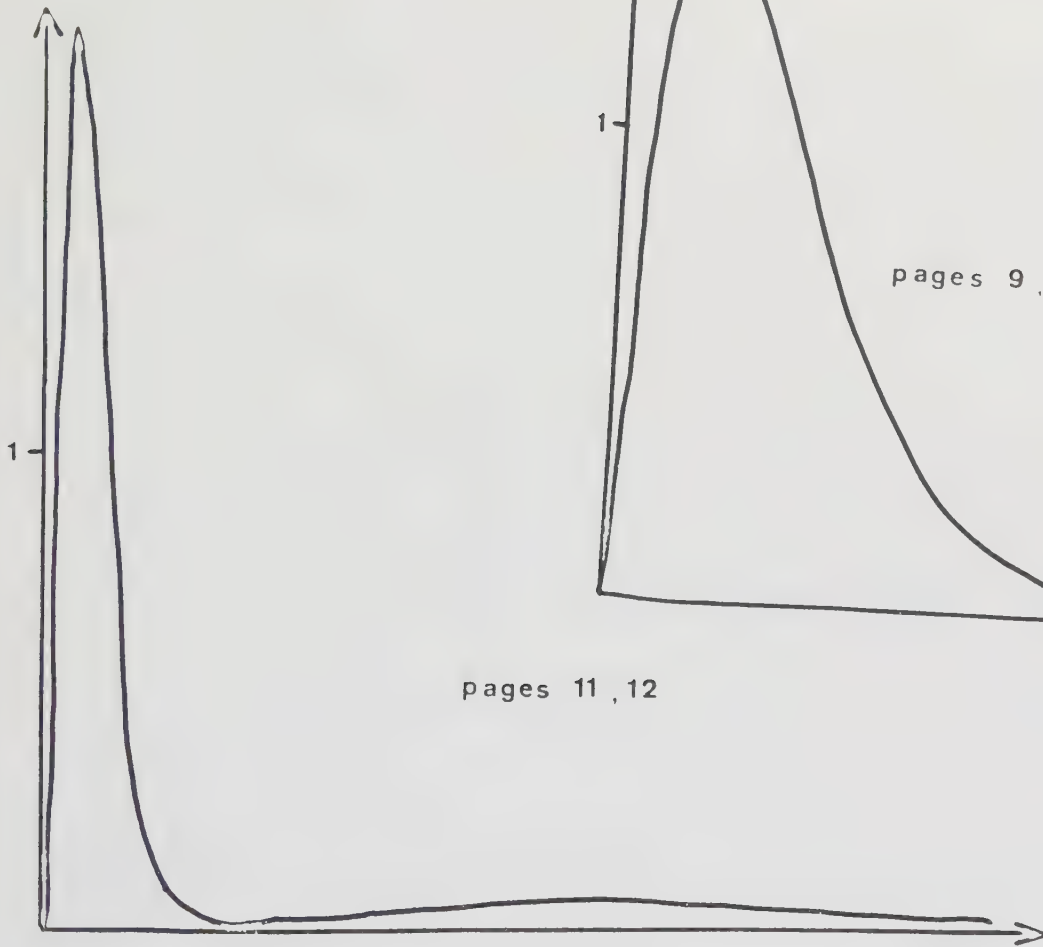


FIGURE 7



$T = 1$

k	1	2	3	4	5	6	7	8
$\Pr \left(w(T) = k \right)$	0,000	0,003	0,020	0,068	0,148	0,218	0,226	0,170
Approx. gaus-sienne	0,001	0,006	0,023	0,067	0,141	0,212	0,228	0,175

k	9	10	11	12	13
$\Pr \left(w(T) = k \right)$	0,094	0,038	0,011	0,002	0,000
Approx. gaus-sienne	0,097	0,038	0,011	0,002	0,000

$T = 4$

k	5	6	7	8	9	10	11	12
$\Pr \left(w(T) = k \right)$	0,000	0,000	0,003	0,017	0,061	0,145	0,246	0,264
Approx. gaus-sienne	0,000	0,000	0,002	0,015	0,062	0,156	0,251	0,257

k	13	14	15	16
$\Pr \left(w(T) = k \right)$	0,178	0,069	0,013	0,001
Approx. gaus-sienne	0,166	0,068	0,018	0,003

T = 9

k	7	8	9	10	11	12	13	14
$\Pr \left(w(T) = k \right)$	0,000	0,000	0,001	0,011	0,067	0,207	0,340	0,276
Approx.gaus-sienne	0,000	0,000	0,001	0,010	0,069	0,219	0,338	0,253

k	15	16	17
$\Pr \left(w(T) = k \right)$	0,088	0,001	
Approx.gaus-sienne	0,092	0,016	0,001

Taille de l'ensemble de travail

FIGURE 8

- . pour $i = 9$ ou 10 : $q_i(u) = 4 u e^{-2u}$ (distribution d'Erlang d'ordre 2)
- . pour $i = 11$ ou 12 : $q_i(u) = (349,98 u^2 - 1,381 \times 10^{-2} u - 2,908 \times 10^{-3}) e^{-10u} + (2,187 \times 10^{-2} u^2 - 1,3812 \times 10^{-2} u + 2,908 \times 10^{-3}) e^{-0,5u}$
- . pour $i = 13$ ou 14 : $q_i(u) = (0,325 u^2 + 0,25) e^{-u}$
- . pour $i = 15$ ou 16 : $q_i(u) = (0,125 u^3 + 0,125 u^2) e^{-u}$

Les coefficients μ_{ij} et a_{ij} de la décomposition de Cox de ces densités sont présentés sur la figure 5. Ces quelques exemples donnent une idée de la diversité de la classe de distributions considérées (voir fig. 7). En particulier le coefficient de variation $\sigma(Q_i) / \bar{Q}_i$ peut prendre des valeurs très variées : la détermination des bornes de ce coefficient de variation pour la sous-classe des distributions admettant une décomposition en r étapes est encore, à notre connaissance, un problème ouvert.

Notons que le coefficient de variation d'une distribution d'Erlang d'ordre k est égal à $1/\sqrt{k}$: ces distributions permettent donc d'approximer les distributions ponctuelles.

La figure 8 montre la distribution de la taille $w(T)$ de l'ensemble de travail pour plusieurs valeurs de la fenêtre. Comme au chapitre 4, cette distribution est comparée avec son approximation gaussienne.

6 - UNE EXTENSION DU MODELE

6.1. DESCRIPTION

L'étude d'un programme réel montre l'importance des corrélations entre pages. Il est fréquent que l'apparition d'une page i dans la localité courante soit suivie à court terme de celle d'une page j , cela pour plusieurs raisons qui ont déjà été signalées (boucle à cheval sur deux pages, appel de sous-programme, séparation du code et des données,...). La prise en compte de cette caractéristique est importante pour l'analyse de la gestion des mémoires virtuelles, en particulier pour l'optimisation, la taille des pages ou la restructuration des programmes. Le modèle étudié ci-dessus ignore ce comportement des programmes (il en est de même des autres modèles cités au chapitre 1, à l'exception toutefois des modèles markoviens d'ordre au moins égal à 1). Nous allons donc en proposer une extension.

Supposons donc que l'entrée dans la localité courante de la page i provoque celle de la page j . On peut modéliser cette caractéristique en introduisant la possibilité pour ces deux pages de partager certaines étapes. On obtient alors un modèle à files d'attente analogue à celui qui est représenté sur la figure 9. Comme dans le modèle présenté au début du chapitre, seule la page i passe par les étapes $u_1, u_2, \dots, u_{r_i}$; après un séjour, dont la durée est distribuée exponentiellement, en u_k , elle aborde l'étape suivante ou entre dans l'état "exécution". Il en est de même pour la page j et les étapes $v_1, v_2, \dots, v_{r_j}$. La nouveauté ce sont les étapes w_1 et w_2 communes aux pages i et j . Ces étapes sont analogues à des serveurs d'un système d'attente qui seraient gérés selon la règle Premier Arrivé - Premier Servi. Lorsque la page i (resp. j) aborde l'étape w_k elle attend, le cas échéant,

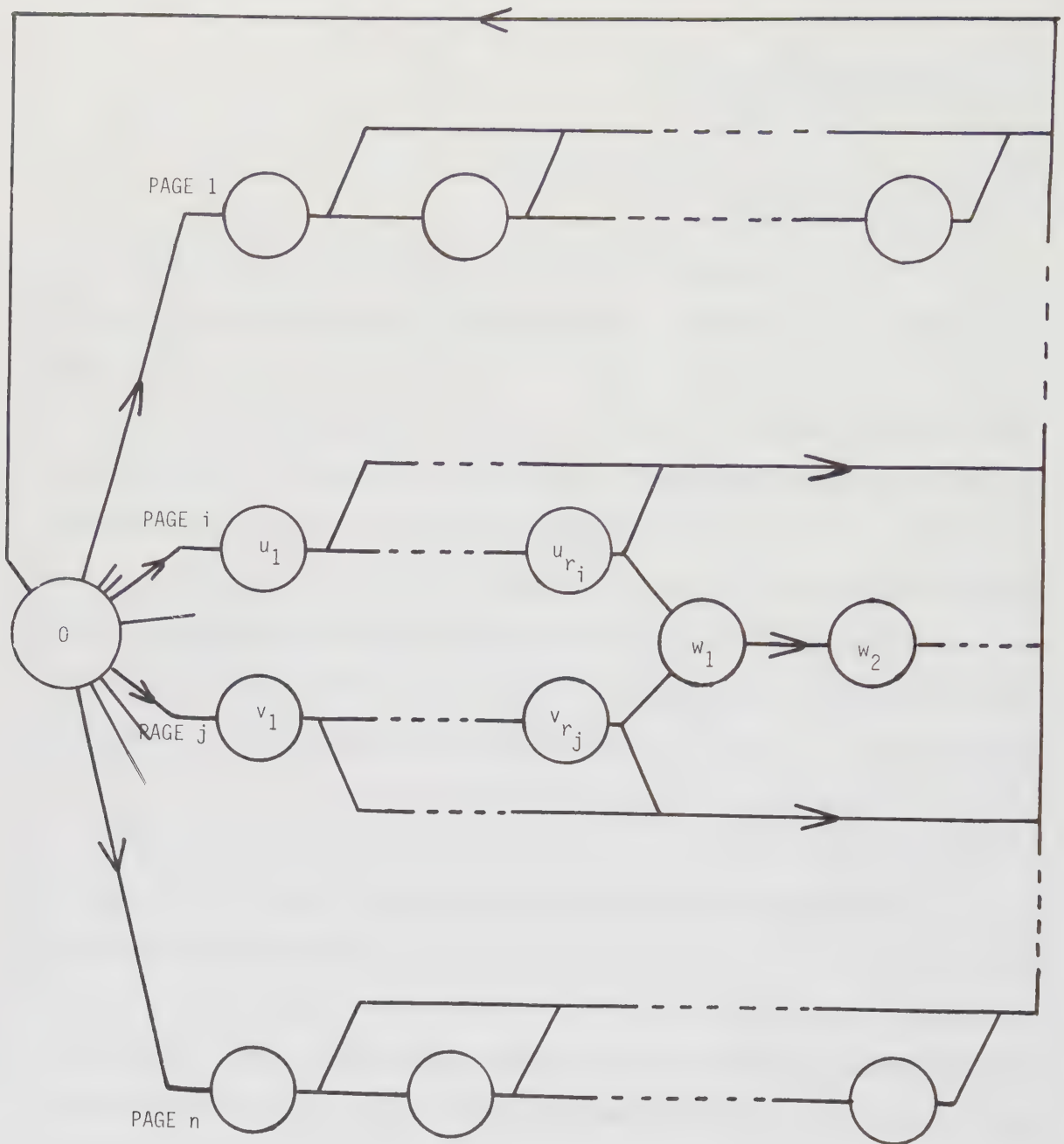


FIGURE 9

que la page j (resp. i) ait quitté cette étape ; elle y entame alors son propre séjour dont la durée a une distribution exponentielle de paramètre $1/\mu_{ki}$ (resp. $1/\mu_{kj}$) à l'issue duquel elle passe elle-même à l'étape suivante. L'état 0 a la même interprétation que précédemment ; en particulier il y a toujours une page et une seule dans cet état.

Le comportement des pages i et j est représenté par les étapes $u_1, u_2, \dots, v_1, v_2, \dots$ lorsque ces pages appartiennent à la localité courante. En revanche les étapes $w_1, w_2, \dots$ correspondent à des temps de séjour plus long. Si les durées moyennes $1/\mu_{1j}, 1/\mu_{2j}, \dots$ sont beaucoup plus faibles que $1/\mu_{1i}, 1/\mu_{2i}, \dots$, on atteint le but recherché. En effet

. si les pages i et j sont toutes deux dans les étapes communes, la page i étant "en tête", le temps de suspension des deux pages est imposé par le comportement de la page i ; si celle-ci entre dans la localité courante (par l'état 0), elle y est suivie peu après par la page j en raison de la relation entre les coefficients des distributions exponentielles

. lorsque la page j est seule dans les étapes communes, elle appartient encore à la localité courante puisque la probabilité qu'elle soit utile à brève échéance est élevée.

On peut envisager plusieurs extensions du modèle originel pour décrire la corrélation des pages. En ce qui concerne leur traitement analytique, toutes sont tributaires de progrès dans la théorie des processus stochastiques. L'extension que nous avons choisi de présenter se place dans le cadre des réseaux de files d'attente à plusieurs classes d'utilisateurs et à serveurs

exponentiels obéissant à la règle P A P S ; cette classe de systèmes est actuellement l'objet de nombreux travaux, ce qui a motivé en partie notre choix. Nous allons maintenant présenter un cas particulier de ces modèles pour lequel nous avons obtenu une expression de la distribution stationnaire.

6.2. UN RESULTAT ANALYTIQUE

Considérons le modèle de programme représenté sur la figure 10. Les n pages sont réparties en g groupes disjoints $\{N_\ell\}_{\ell=1,2,\dots,g}$. Désignons par n_ℓ le nombre d'éléments de N_ℓ et posons

$$v_\ell = \sum_{i=1}^{\ell-1} n_i.$$

La numérotation des pages est supposée telle que N_ℓ se compose des pages $v_\ell + 1, v_\ell + 2, \dots, v_{\ell+1}$.

Si la page i du groupe N_ℓ n'est pas en train d'être référencée, elle est en attente à la station ℓ où les pages sont classées selon leur ordre d'arrivée. La configuration du modèle à un instant donné peut donc être décrite par n -uplet $\sigma = \left[s_1, s_2, \dots, s_i, \dots, s_n \right]$ tel que

- si la page i est en train d'être référencée, $s_i = 0$
- sinon, s_i est le rang de la page i à la station ℓ

À l'instant 0, l'état du programme est un n -uplet

$$\sigma^{(0)} = \left[s_1^{(0)}, s_2^{(0)}, \dots, s_i^{(0)}, \dots, s_n^{(0)} \right]$$

Cet état est possible ce qui implique que l'application ρ_ℓ qui à un élément i de N_ℓ associe l'entier $s_i^{(0)}$ est une bijection sur $\{1, 2, \dots, n_\ell\}$

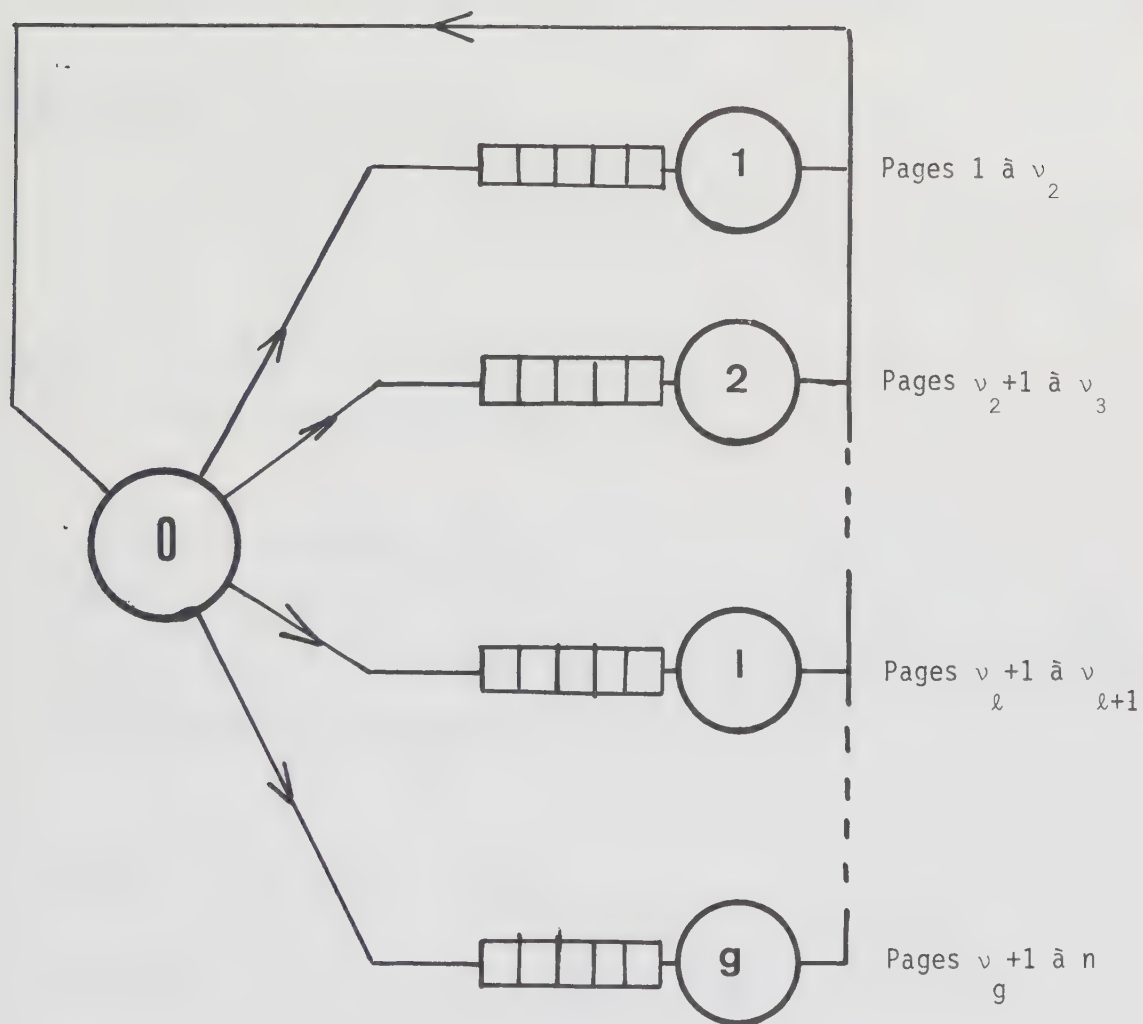


FIGURE 10

sauf pour un indice ℓ_0 ($1 \leq \ell_0 \leq g$) où elle est une bijection sur $\{0, 1, 2, \dots, n_{\ell_0} - 1\}$. Le programme évolue de la manière suivante :

- la durée qui s'écoule entre l'instant où la page i parvient au premier rang de la station ℓ et l'instant où elle quitte cette station a une distribution exponentielle de paramètre μ_i .

- la page qui occupe l'état 0 y reste jusqu'à ce qu'elle soit remplacée par une autre page. Elle rejoint alors la station de son groupe.

Remarquons que l'ordre initial des pages d'un groupe ne peut être complètement bouleversé. Nous allons préciser l'ensemble des états possibles du système. Pour tout ℓ ($1 \leq \ell \leq g$), soit $\mathcal{F}^{(\ell)}$ la famille d'applications de N_ℓ dans $\{0, 1, \dots, n_\ell\}$ définie récursivement par les règles suivantes :

- . l'application ρ_ℓ appartient à $\mathcal{F}^{(\ell)}$
- . si f est une application de $\mathcal{F}^{(\ell)}$ telle que l'ensemble $f(N_\ell)$ ne contienne pas 0, alors $f - 1$ appartient également à $\mathcal{F}^{(\ell)}$.
- . si f est une application de $\mathcal{F}^{(\ell)}$ telle que $f(N_\ell)$ contienne 0 et si τ désigne la transposition sur $\{0, 1, \dots, n_\ell\}$ qui échange 0 et n_ℓ alors $\tau \circ f$ appartient également à $\mathcal{F}^{(\ell)}$.

On vérifie aisément que l'ensemble S des états possibles du système est l'ensemble des n -uples $\sigma = (s_1, s_2, \dots, s_1, \dots, s_n)$ tels qu'il existe une famille d'applications $(f_\ell)_{1 \leq \ell \leq g}$ ($f_\ell \in \mathcal{F}^{(\ell)}$) satisfaisant à la condition :

$$(\forall \ell) (1 \leq \ell \leq g) \quad \left(\forall i \in N_\ell \right) \quad f_\ell(i) = s_i$$

La distribution limite de ce système fait l'objet du théorème ci-dessous

Théorème 4 A tout état possible $\sigma = (s_1, s_2, \dots, s_n)$ associons les deux parties $E(\sigma)$ et $F(\sigma)$ de N définies par

. $i \in E(\sigma) \iff (s_i = 1 \text{ et le groupe } N_\ell \text{ auquel appartient } i \text{ ne contient pas la page dans l'état } 0)$

. $i \in F(\sigma) \iff \left(s_i = 1 \text{ ou } (i \in N_\ell \text{ et } n_\ell = 1 \text{ et } g_i = 0) \right)$

La probabilité de σ dans la distribution stationnaire du système est

$$(47) \quad \pi(\sigma) = \frac{K}{\prod_{i \in E(\sigma)} \mu_i \sum_{i \in F(\sigma)} \mu_i}$$

où K désigne la constante de normalisation :

$$K = \left(\sum_{\sigma \in S} \frac{1}{\prod_{i \in E(\sigma)} \mu_i \sum_{i \in F(\sigma)} \mu_i} \right)^{-1}$$

Ce résultat est établi dans l'annexe B

ANNEXE A

DEMONSTRATION DU THEOREME 1

Remarquons que ce théorème est trivialement vrai pour des programmes à une seule page. Nous allons calculer la probabilité que le programme -en équilibre statistique- soit dans l'état σ à l'instant t et que, pour tout i différent de $c(\sigma)$, on ait : $u_i \leq U_i < u_i + du_i$

Pour cela nous supposons

$$(A1) \quad c(\sigma) = 1$$

$$(A2) \quad u_2 < u_3 < \dots < u_n$$

Ces deux hypothèses ne restreignent pas la généralité de la démonstration : puisque l'évènement $U_i = U_j$ ($i \neq j$) a une probabilité nulle, il suffit de renuméroter les pages, éventuellement, pour que les relations A1 et A2 soient satisfaites.

Le théorème va être démontré par récurrence sur n . Nous utiliserons les notations ci-dessous :

. $\mathcal{P}^{(n)}$ est le système qui a été utilisé jusqu'ici pour modéliser le comportement du programme.

. $\mathcal{P}^{(n-1)}$ est dérivé du système $\mathcal{P}^{(n)}$ par suppression de la page n ; les constantes r_i, μ_{ij}, a_{ij} relatives aux pages 1 à $n-1$ sont les mêmes que dans $\mathcal{P}^{(n)}$.

. De la même manière, on obtient $\mathcal{P}^{(i)}$ ($2 \leq i \leq n$) en restreignant $\mathcal{P}^{(n)}$ à ses i premières pages.

. $\mathcal{L}^{(n)}$ est le réseau représentant la décomposition de Cox de la variable aléatoire Q_n (temps de suspension de la page n)

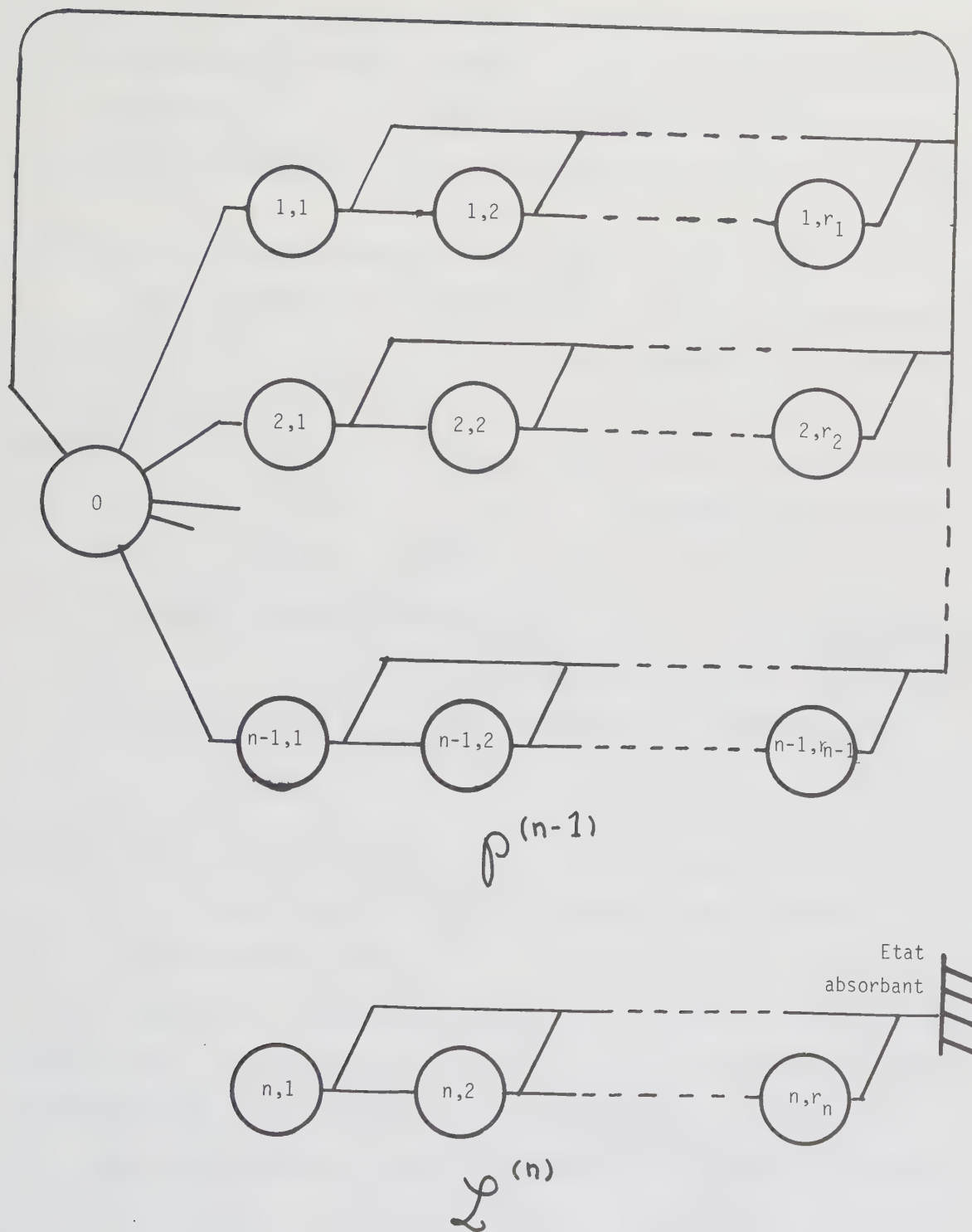


FIGURE 11

. Les éléments relatifs à $\mathcal{P}^{(i)}$ sont distingués par un indice supérieur i entre parenthèses. Par exemple, l'ensemble d'états et la distribution stationnaire de $\mathcal{P}^{(i)}$ sont notés respectivement $S^{(i)}$ et $\pi^{(i)}$. Les symboles $S^{(n)}$ et $\pi^{(n)}$ sont donc équivalents aux notations S et π utilisées jusqu'ici.

Le fil directeur de la démonstration est de considérer le système $\mathcal{P}^{(n)}$ (représenté sur la figure 3) comme équivalent du couple $(\mathcal{P}^{(n-1)}, \mathcal{L}^{(n)})$ (représenté sur la figure 11) durant l'intervalle $]t-u_n, t]$. La principale difficulté est de s'assurer que l'on manipule toujours des distributions stationnaires (voir le lemme 2). Pour expliciter les conditions initiales, nous introduisons deux notations :

. le système $\mathcal{P}^{(i)}$ sera désigné par $\mathcal{P}^{(i)}_{(\tau_-)}$ (resp. $\mathcal{P}^{(i)}_{(\tau_+)}$) s'il existe un réel $h > 0$ tel que la distribution de l'état du système soit la distribution d'équilibre durant $]\tau-h, \tau[$ (resp. $]\tau, \tau+h[$)

. le système $\mathcal{L}^{(i)}$ sera désigné par $\mathcal{L}^{(i)}_{(\tau)}$ s'il est dans l'état 1 à l'instant τ .

Comme nous allons le montrer, si la page n quitte l'état 0 à l'instant τ , la probabilité que le système $\mathcal{P}^{(n)}_{(\tau_-)}$ soit dans l'état $(j_1, \dots, j_{n-1}, 1)$ juste après cette transition est égale à la probabilité stationnaire que $\mathcal{P}^{(n-1)}$ soit dans l'état $(j_2, \dots, j_{n-1})$. Par conséquent, si la page n quitte l'état 0 à l'instant $t-u_n$ et n'y revient pas durant $]t-u_n, t]$ (cette condition sera notée C), le comportement du système $\mathcal{P}^{(n)}_{(t-u_n-)}$ est équivalent à celui de la paire $(\mathcal{P}^{(n-1)}_{(t-u_n+)}, \mathcal{L}^{(n)}_{(t-u_n)})$ au moins durant $]t-u_n, t]$; "équivalent" signifie ici :

$$(\forall t' \in]t-u_n, t]) (\forall (j_1, j_2, \dots, j_n) \in S^{(n)}) (j_n \neq 0) :$$

$$\Pr[\mathcal{P}^{(n)}_{(t-u_n-)} \text{ dans l'état } (j_1, j_2, \dots, j_n) \text{ à l'instant } t' \mid C] =$$

$$\Pr[\mathcal{P}^{(n-1)}_{(t-u_n+)} \text{ dans l'état } (j_1, j_2, \dots, j_{n-1}) \text{ à l'instant } t'] \times$$

$$\Pr[\mathcal{L}^{(n)}_{(t-u_n)} \text{ dans l'état } j_n \text{ à l'instant } t' \mid \mathcal{L}^{(n)}_{(t-u_n)} \text{ n'est pas absorbé durant }]t-u_n, t]]$$

Lemme 1

Soit $(V_i)_{i=1, \dots, \ell}$ une famille de ℓ variables aléatoires indépendantes de paramètres $(\mu_i)_{i=1, \dots, \ell}$; alors

$$\Pr [V_1 = \inf_i (V_i)] = \frac{\mu_1}{\sum_{i=1}^{\ell} \mu_i}$$

Démonstration

Puisque les variables aléatoires sont indépendantes,

$$\begin{aligned} \Pr [V_1 = \inf_i (V_i)] &= \int_{s=0}^{\infty} \Pr [s-ds < V_1 \leq s] \times \Pr [\text{pour tout } i \geq 2 : V_i > s] \\ &= \int_{s=0}^{\infty} \mu_1 e^{-\mu_1 s} e^{-\mu_2 s} e^{-\mu_3 s} \dots e^{-\mu_{\ell} s} ds \\ &= \frac{\mu_1}{\sum_{i=1}^{\ell} \mu_i} \end{aligned}$$

Lemme 2

Soit $(j_1, j_2, \dots, j_{n-1}, 1)$ un état de $\mathcal{P}^{(n)}$. Si la page n quitte l'état 0 à l'instant τ , la probabilité (notée $\psi(j_1, j_2, \dots, j_{n-1})$) que le système $\mathcal{P}^{(n)}_{(\tau-)}$

soit dans l'état $(j_1, j_2, \dots, j_{n-1}, 1)$ juste après cette transition est égale à $\pi^{(n-1)}(j_1, j_2, \dots, j_{n-1})$.

Démonstration

a) Au processus en temps continu $P^{(n)}$ associons la chaîne de Markov incluse $\mathcal{M}$ de ses états successifs :

- l'ensemble des états de $\mathcal{M}$ est $S^{(n)}$
- $\mathcal{M}$ est dans l'état σ après sa q^e transition si $P^{(n)}$ est dans cet état après sa propre q^e transition.

Cette chaîne de Markov est ergodique : pour tout $\sigma \in S^{(n)}$, soit $\phi(\sigma)$ la probabilité stationnaire de σ considéré comme état de $\mathcal{M}$. En comparant les équations d'équilibre de $P^{(n)}$ et $\mathcal{M}$, nous vérifions aisément que

$$(A3) \quad \forall \sigma = (j_1, j_2, \dots, j_n) \in S^{(n)} : \phi(\sigma) = R \pi^{(n)}(\sigma) \sum_{\substack{i=1 \\ i \neq c(\sigma)}}^n \mu_{ij_i}$$

où R désigne une constante indépendante de σ .

b) Notons J^* l'évènement

" La dernière transition qui s'est produite avant τ a fait passer la page n de l'état 0 à l'état 1."

Soit $\sigma = (j_1, j_2, \dots, j_{n-1}, 1)$ un état de $P^{(n)}$: nous le noterons $(\sigma', 1)$ pour attirer l'attention sur $\sigma' = (j_1, j_2, \dots, j_{n-1})$ qui est un état de $P^{(n-1)}$. Il est clair que

$$\begin{aligned}
 \psi(j_1, j_2, \dots, j_{n-1}) &= \Pr[\mathcal{M} \text{ est dans l'état } (\sigma', 1) \mid J^*] \\
 (A4) \qquad \qquad \qquad &= \frac{\Pr[\mathcal{M} \text{ est dans l'état } (\sigma', 1) \text{ et } J^*]}{\Pr[J^*]}
 \end{aligned}$$

Calculons séparément chacune des probabilités qui apparaissent dans le rapport ci-dessus. L'évènement ($\mathcal{M}$ est dans l'état $(\sigma', 1)$ et J^*) se produit si et seulement si les trois éventualités suivantes sont réalisées :

- juste avant la dernière transition $\mathcal{P}^{(n)}$ était dans un état de la forme

$$(j_1, j_2, \dots, j_{c(\sigma)-1}, \ell, j_{c(\sigma)+1}, \dots, j_{n-1}, 0) = r_{n\ell}(\sigma) \quad (1 \leq \ell \leq r_{c(\sigma)})$$

- la page $c(\sigma)$ est la première à changer d'état et cause donc la "dernière transition". La probabilité de cet évènement est donnée par le lemme 1.

- la page $c(\sigma)$ passe à l'état 0

Par conséquent :

$$(A5) \quad \Pr[\mathcal{M} \text{ dans l'état } (\sigma', 1) \text{ et } J^*] = \sum_{\ell=1}^{r_k} \phi(r_{n\ell}(\sigma)) \frac{\mu_{k\ell}}{\mu_{k\ell} + \sum_{\substack{i=1 \\ i \neq k}}^{n-1} \mu_{ij_i}} b_{k\ell}$$

où k désigne l'entier $c(\sigma)$.

De (A3) il s'ensuit que cette probabilité est égale à

$$\sum_{\ell=1}^{r_k} R \pi^{(n)}(\Gamma_{n\ell}(\sigma)) \mu_{k\ell} b_{k\ell}$$

Conformément à l'égalité (8) cette somme est égale à

$$\sum_{\ell=1}^{r_k} R \pi^{(n)}(\sigma) A_{k\ell} b_{k\ell} \mu_{n1} = R \pi^{(n)}(\sigma) \mu_{n1} \sum_{\ell=1}^{r_k} A_{k\ell} b_{k\ell}$$

La sommation de droite étant égale à 1 (d'après l'égalité 2), nous en déduisons à l'aide de la formule (5)

$$(A6) \quad \Pr [\mathcal{M} \text{ est dans l'état } \sigma = (\sigma', 1) \text{ et } J^*] = R \frac{G^{(n)}}{G^{(n-1)}} \pi^{(n-1)}(\sigma')$$

En ce qui concerne l'évènement J^* , sa probabilité est maintenant facile à calculer :

$$(A7) \quad \begin{aligned} \Pr [J^*] &= \sum_{\sigma'} \Pr [\mathcal{M} \text{ est dans l'état } (\sigma', 1) \text{ et } J^*] \\ &= R \frac{G^{(n)}}{G^{(n-1)}} \end{aligned}$$

En rapprochant les égalités (A6) et (A7), nous obtenons

$$\psi(\sigma') = \pi^{(n-1)}(\sigma')$$

ce qui achève la démonstration du lemme 2

Démonstration du théorème 1

Soit $\sigma = (0, j_2, \dots, j_{n-1}, j_n)$ un état possible de $\mathcal{P}^{(n)}$; soient $u_2, u_3, \dots, u_{n-1}, u_n$ des nombres réels tels que

$$0 < u_2 < u_3 \dots < u_{n-1} < u_n$$

En accord avec les notations introduites au début de cet appendice, nous écrirons $\eta^{(n)}$ à la place de η pour expliciter le fait que cette distribution de probabilité est relative au système $\mathcal{P}^{(n)}$. De manière évidente :

$$\begin{aligned} \eta^{(n)}(\sigma ; u_2, u_3, \dots, u_{n-1}, u_n) du_2 du_3 \dots du_{n-1} du_n = \\ \Pr [A \text{ l'instant } t, \mathcal{P}^{(n)}(t-u_n-) \text{ est dans l'état } \sigma \text{ et } u_i - du_i < U_i \leq u_i \\ \text{pour tout } 2 \leq i \leq n] \\ = \mathcal{A} \times \mathcal{B} \times \mathcal{C} \end{aligned}$$

où nous avons posé

$$\begin{aligned} \mathcal{A} = \Pr [\text{à l'instant } t, \mathcal{P}^{(n)}(t-u_n-) \text{ est dans l'état } \sigma \text{ et} \\ u_i - du_i < U_i \leq u_i + du_i \text{ pour tout } i \mid \mathcal{E}] \end{aligned}$$

$\mathcal{E}$ désignant l'évènement :

" la page n quitte l'état 0 au cours de l'intervalle
 $]t-u_n, t-u_n + du_n[$, n'y rentre pas avant t et se trouve dans
l'état j_n à l'instant t "

$$\begin{aligned} \mathcal{B} = \Pr [\text{la page } n, \text{ du système } \mathcal{P}^{(n)}(t-u_n-), \text{ quitte l'état } 0 \text{ entre } t-u_n \\ \text{et } t-u_n + du_n] \end{aligned}$$

$\mathcal{C} = \text{Pr} \left[\text{dans l'intervalle de temps }]t - u_n, t[, \text{ la page } n, \text{ partant de l'état } 1, \text{ atteint l'état } j_n \text{ sans repasser par l'état } 0 \right]$

Puisque l'évènement (à l'instant $t : J_n = j_n$ et $u_n - du_n < U_n \leq u_n$) est inclus dans $\mathcal{E}$, nous pouvons écrire

$$\mathcal{C} = \text{Pr} \left[\text{à l'instant } t, \text{ dans le système } \mathcal{P}^{(n)}(t - u_n), \text{ pour tout } \right. \\ \left. 2 \leq i \leq n-1, J_i = j_i \text{ et } u_i - du_i < U_i \leq u_i ; J_1 = 0 \mid \mathcal{E} \right]$$

D'après le lemme 2 et puisque le comportement de la page n n'affecte pas les autres pages durant $]t - u_n, t[$:

$$\mathcal{C} = \text{Pr} \left[\text{à l'instant } t, \mathcal{P}^{(n-1)}(t - u_n) \text{ est dans l'état } \sigma' = (0, j_2, \dots, j_{n-1}) \right. \\ \left. \text{et } u_i - du_i < U_i \leq u_i \text{ pour } 2 \leq i \leq n-1 \right]$$

Cette expression de $\mathcal{C}$ est équivalente à

$$(A9) \quad \mathcal{C} = \pi^{(n-1)}(\sigma' ; u_2, u_3, \dots, u_{n-1}) du_2 du_3 \dots du_{n-1}$$

Il est à noter que $\mathcal{C}$ est maintenant exprimé comme une probabilité absolue relative à $\mathcal{P}^{(n-1)}$: Intéressons nous maintenant à $\mathcal{B}$:

$$\mathcal{B} = \sum_{*} \pi^{(n)}(l_1, l_2, \dots, l_{n-1}, 0) \sum_{i=1}^{n-1} u_{i, l_i} b_{i, l_i} du_n$$

où $\sum_{*}$ désigne une sommation sur l'ensemble des éléments $(l_1, l_2, \dots, l_{n-1}, 0)$ de $S^{(n)}$ dont la dernière composante est nulle.

Il résulte de la proposition 1 :

$$\mathcal{B} = G^{(n)} \sum_{*q=1}^{n-1} \frac{A_{q, l_q}}{\pi_{q, l_q}} \sum_{i=1}^{n-1} u_{i, l_i} b_{i, l_i} du_n$$

$$\beta = G^{(n)} \sum_{*} \sum_{i=1}^{n-1} \prod_{\substack{q=1 \\ q \neq i}}^{n-1} \frac{A_{q\ell_q}}{\pi_{q\ell_q}} A_{i\ell_i} b_{i\ell_i} du_n$$

Permutons les deux sommations ; puis, en effectuant $\sum_{*}$, groupons les n-uples $(\ell_1, \ell_2, \dots, \ell_{n-1}, 0)$ qui diffèrent uniquement par leur i^e composante de sorte que nous puissions utiliser l'égalité (2).

Nous obtenons :

$$\beta = G^{(n)} \sum_{i=1}^{n-1} \sum_{*} \prod_{\substack{q=1 \\ q \neq i}}^{n-1} \frac{A_{q\ell_q}}{\pi_{q\ell_q}} du_n$$

c'est à dire :

$$(A10) \quad \beta = \frac{G^{(n)}}{G^{(n-1)}} du_n$$

Intéressons nous enfin, à $\mathcal{C}$:

$$\mathcal{C} = A_{nj_n} H_{nj_n} (u_n)$$

où H_{nj_n} désigne la fonction définie dans la formule (13).

Conformément à (15) :

$$(A11) \quad \mathcal{C} = \frac{A_{n,j_n}}{\mu_{n,j_n}} e_{nj_n} (u_n)$$

Rassemblons les égalités (A8) (A9) (A10) (A11) :

$$\eta^{(n)}(\sigma; u_2, u_3, \dots, u_n) du_2 du_3 \dots du_n = \left(\frac{G^{(n)}}{G^{(n-1)}} \eta^{(n-1)}(\sigma'; u_2, u_3 \dots u_{n-1}) \right. \\ \left. du_2 du_3 \dots du_{n-1} \right) \frac{A_{nj_n}}{\mu_{nj_n}} e_{nj_n}(u_n) du_n$$

où $\sigma = (\sigma', j_n)$.

Cette relation de récurrence sur n étant établie, la démonstration s'achève immédiatement puisque le théorème est trivialement vrai pour des programmes à une seule page.

ANNEXE B

DEMONSTRATION DU THEOREME 4 ---

Afin de ne pas alourdir exagérément les notations, nous n'écrirons que les équations d'équilibre relatives à des états du programme où la page en cours d'utilisation est la page 1.

Soit $\sigma = (0, s_2, s_3, \dots, s_n)$ un tel état du programme.

Soit t un entier compris entre 1 et g : si l'entrée dans l'état 0 de la page 1 en chasse une page de N_t , le programme passe dans l'état σ à condition qu'il ait été dans un état $H_t(\sigma) = (s'_1, s'_2, \dots, s'_i, \dots, s'_n)$ ainsi défini :

- si $t = 1$:
 - $s'_i = s_i + 1$ si i appartient à N_1 et que $s_i \neq n_1 - 1$
 - $s'_i = 0$ si i appartient à N_1 et que $s_i = n_1 - 1$
 - $s'_i = s_i$ sinon
- si $t > 1$:
 - $s'_i = s_i + 1$ si i appartient à N_1
 - $s'_i = 0$ si i appartient à N_t et que $s_i = n_i$
 - $s'_i = s_i$ sinon

Pour tout ℓ ($1 \leq \ell \leq g$) désignons par ϕ_ℓ la page de N_ℓ telle que $s_{\phi_\ell} = 1$.

Pour l'écriture des équations d'équilibre, nous devons distinguer deux cas :

Premier cas : $n_1 > 1$

L'équation est alors :

$$(B1) \quad \pi(\sigma) \sum_{\ell=1}^g \mu_{\phi_\ell} = \mu_1 \sum_{t=1}^g \pi(H_t(\sigma))$$

D'après le théorème 4, les probabilités stationnaires figurant ci-dessus seraient respectivement :

$$\bullet \quad \pi(\sigma) = \frac{K}{\prod_{\ell=2}^g \mu_{\phi_\ell} \times \prod_{\ell=1}^g \mu_{\phi_\ell}}$$

$$\bullet \quad \pi(H_1(\sigma)) = \frac{K}{\prod_{\ell=2}^g \mu_{\phi_\ell} \times (\mu_1 + \sum_{\ell=2}^g \mu_{\phi_\ell})} = \frac{K \mu_1}{D}$$

$$\text{où l'on a posé } D = \mu_1 \prod_{\ell=2}^g \mu_{\phi_\ell} \times (\mu_1 + \sum_{\ell=2}^g \mu_{\phi_\ell})$$

$$\bullet \quad \pi(H_t(\sigma)) = \frac{K}{\prod_{\substack{\ell=2 \\ \ell \neq t}}^g \mu_{\phi_\ell} (\mu_1 + \sum_{\ell=2}^g \mu_{\phi_\ell})} = \frac{K \mu_{\phi_t}}{D} \quad (t > 1)$$

Nous pouvons donc écrire :

$$\mu_1 \sum_{t=1}^g \pi(H_t(\sigma)) = \frac{K \mu_1 (\mu_1 + \sum_{t=2}^g \mu_{\phi_t})}{D}$$

$$= \frac{K}{\prod_{\ell=2}^g \mu_{\phi_\ell}}$$

$$= \pi(\sigma) \sum_{\ell=1}^g \mu_{\phi_\ell}$$

Ce qu'il fallait vérifier.

cas : $\mu_1 = 1$

L'équation d'équilibre est

$$(B2) \quad \pi(\sigma) \sum_{\phi=2}^g \mu_{\phi} = \mu_1 \sum_{t=2}^g \pi(H_t(\sigma))$$

Les probabilités proposées dans le théorème 4 sont

$$\pi(\sigma) = \frac{K}{\mu_{\phi_2} \mu_{\phi_3} \dots \mu_{\phi_g} (\mu_1 + \mu_{\phi_2} + \dots + \mu_{\phi_g})}$$

et

$$\begin{aligned} \pi(H_t(\sigma)) &= \frac{K}{\mu_1 \mu_{\phi_2} \dots \mu_{\phi_{t-1}} \mu_{\phi_{t+1}} \dots \mu_{\phi_g} (\mu_1 + \mu_{\phi_2} + \dots + \mu_{\phi_g})} \\ &= \pi(\sigma) \frac{\mu_{\phi_t}}{\mu_1} \end{aligned}$$

La vérification de l'accord entre cette solution et l'équation B2 est immédiate.

REFERENCES

- [1] LENFANT J. : The delay Network Model of program behaviour - Actes du Séminaire international sur la modélisation des systèmes et réseaux informatiques, Rocquencourt, North Holland P.C. (1974)
- [2] COX D.R. : A use of complex probabilities in the theory of stochastic processes - Proc. Camb. Phil. Soc. , Vol. 51 (1955)
- [3] DENNING P.J. - SCHWARTZ S.C. : Properties of the working set model - Comm. A.C.M., Vol. 15, 3, Mars 1972

GESTION DES MEMOIRES A DISQUE OU A TAMBOUR

1 INTRODUCTION

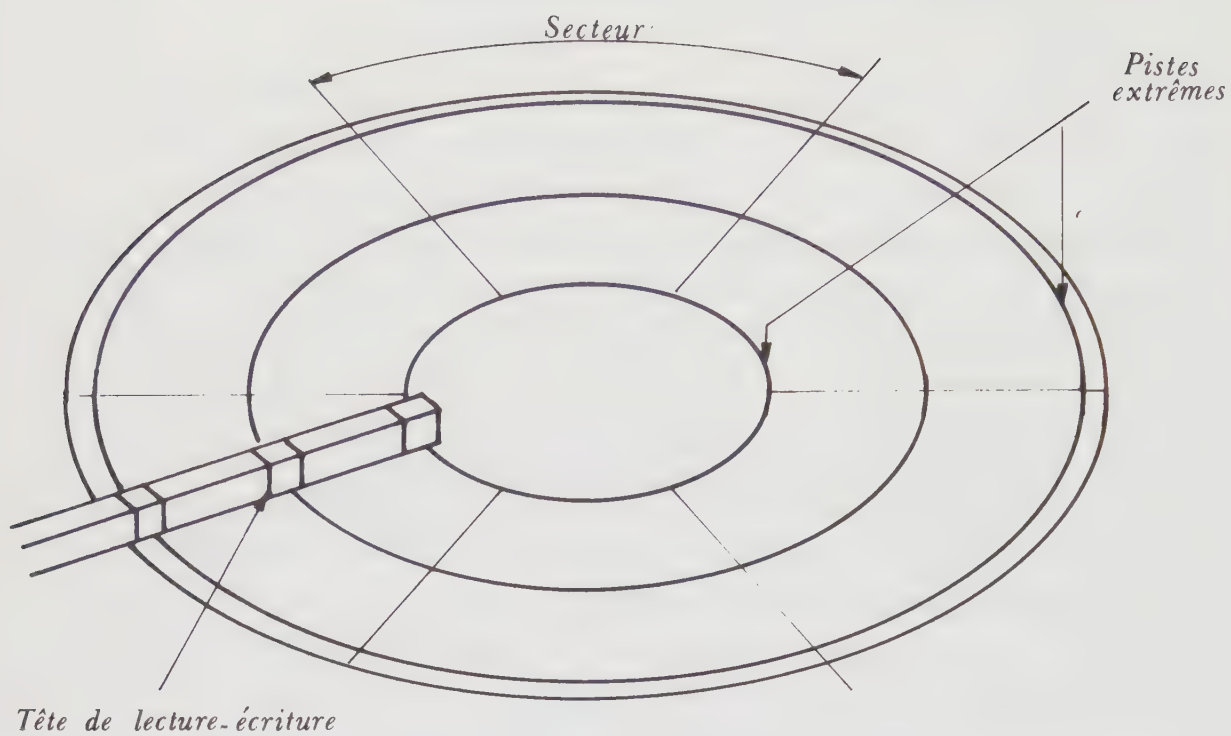
La nécessité de disposer de mémoires de grande capacité explique l'essor des mémoires rotatives à support magnétique (disques, tambours, mémoires à bulles) dont le coût est bien moindre que celui des mémoires à tores ou semi-conducteurs. Ces mémoires rotatives ont en commun une propriété essentielle : le temps d'accès à l'information a une borne supérieure, dépendant uniquement des caractéristiques électro-mécaniques du dispositif, qui est relativement faible : elle est généralement comprise entre quelques millisecondes et quelques centaines de millisecondes. Cette caractéristique autorise l'utilisation de structures de données variées. Elle permet aussi à ces mémoires magnétiques d'être, conjointement avec la mémoire principale, le support physique des mémoires virtuelles ; de ce point de vue elles apparaissent donc comme des extensions de la mémoire principale. A l'opposé, lorsqu'on utilise des bandes magnétiques, la consultation séquentielle de grandes quantités d'information est de rigueur.

Sur un disque, l'information est matérialisée sur les pistes, cercles concentriques centrés sur l'axe de rotation et recouverts d'une substance magnétisable. La lecture et l'écriture se font par des têtes sensibles à l'état de magnétisation de la piste au-dessus de laquelle elles se trouvent. Les disques sont généralement classés en deux catégories :

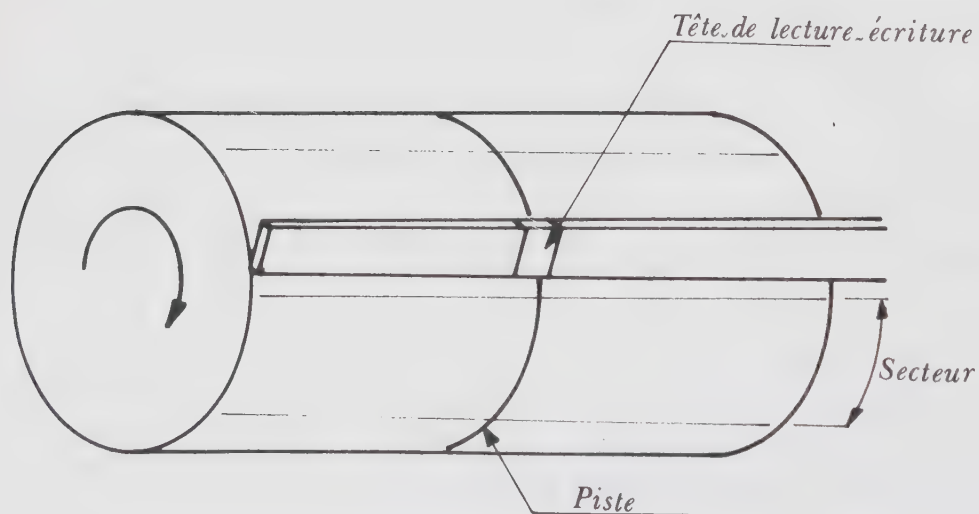
- Les disques à têtes mobiles : les têtes de lecture en nombre inférieur à celui des pistes sont portées par un bras mobile. Lorsqu'un ordre de lecture ou d'écriture commence à être exécuté, le mouvement de ce bras place les têtes au-dessus de la piste concernée : le temps nécessaire à cette opération est bien souvent le facteur principal du temps de réponse du disque.

- Les disques à têtes fixes : au-dessus de chaque piste se trouve une (plus rarement deux) tête de lecture/écriture. Le coût du dispositif est donc plus élevé mais au mouvement du bras est substituée une commutation électronique presque instantanée. A cette catégorie de disques on peut rattacher les tambours.

Ces mémoires magnétiques ayant des temps de réponse élevés par rapport au temps nécessaire à l'exécution d'une instruction, l'attente d'entrée/sortie ne peut pas toujours être satisfaite avant qu'il en sur-



a) Disque à têtes fixes



b) Tambour

Figure 1

vienne une autre : il se forme donc des files d'attente. Nous verrons que l'efficacité de ce type de mémoire dépend très étroitement de l'ordre dans lequel sont servies les demandes qui n'ont encore pu être satisfaites.

Dans les paragraphes qui suivent, nous allons décrire plus en détail les disques et tambours et présenter diverses études consacrées à l'optimisation de leur emploi.

1.1 DISQUES A TÊTES FIXES ET TAMBOUR.

Une des deux faces d'un disque à têtes fixes porte des pistes magnétisées à quelques dizaines de microns au-dessus desquelles sont maintenues les têtes de lecture/écriture. Sur certains disques, chaque piste est divisée en secteurs qui constituent la plus petite quantité d'information adressable (voir figure 1a). Les ordres de transfert envoyés par l'unité centrale ou par les calculateurs "canaux" (encore appelés unités d'échange) sont reçus par un ensemble de circuits logiques, l'unité de liaison du disque. A ce niveau, l'ordre reçu est à exécuter immédiatement : les files d'attente éventuelles ont été traitées par le superviseur d'entrée/sortie. Le disque tournant en permanence, dès que le début du secteur de piste demandé passe sous les têtes de lecture/écriture, le transfert commence ; il se poursuit en parallèle avec la rotation du disque et s'achève lorsque la fin du secteur passe sous les têtes. Une seule tête peut être active à la fois. La cadence de transfert, exprimée en octets par unité de temps, est donc égale à :

$$\frac{\text{Nombre d'octets sur un secteur de piste} \times \text{Nombre de secteurs}}{\text{Période de révolution}}$$

Deux secteurs consécutifs d'une même piste sont séparés par un intervalle de sécurité destiné à fournir le temps nécessaire à une réinitialisation des circuits électroniques entre deux transferts. En outre sur de nombreux modèles de disque, au cours du passage de cet intervalle sous les têtes de lecture celles-ci peuvent être commutées de l'état écriture à l'état lecture ou vice-versa : des transferts en sens opposés mettant en jeu des secteurs consécutifs peuvent donc être programmés sans que cela cause un retard d'une période de rotation lors de l'exécution.

La figure 2 indique quelques caractéristiques des disques DIAD 7204 (Compagnie Internationale pour l'Informatique) qui sont destinés à des cal-

Nombre de pistes	512
Nombre de secteurs	16
Capacité d'un secteur de piste	360 octets
Capacité totale du disque	3 millions d'octets
Vitesse de rotation	1480 tours par minute
Durée d'une révolution	40,56 ms
Durée du passage sous une tête de l'intervalle de sécurité entre deux secteurs	216 μ s
Intervalle de temps entre le passage sous les têtes du début de deux secteurs consécutifs	2,56 ms
Cadence de transfert	142 000 octets par seconde

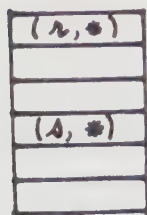
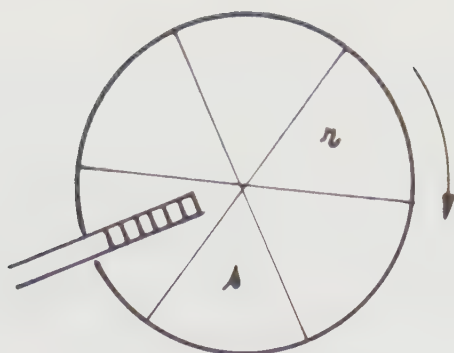
Caractéristiques des DIAD 7204

Figure 2

culateurs dont la mémoire principale a un temps de cycle voisin d'une micro-seconde.

D'un point de vue externe, les tambours sont très proches des disques à têtes fixes. Les pistes sont des cercles sur la surface d'un cylindre. Elles sont divisées en secteurs et disposent chacune d'une tête de lecture/écriture. Les vitesses de rotation peuvent être très grandes (jusqu'à 24 000 tours par minute).

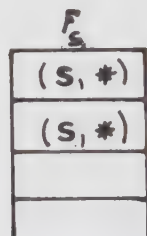
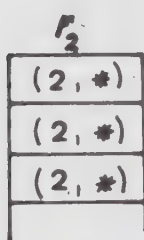
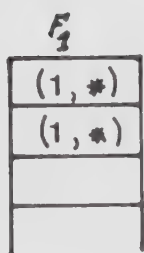
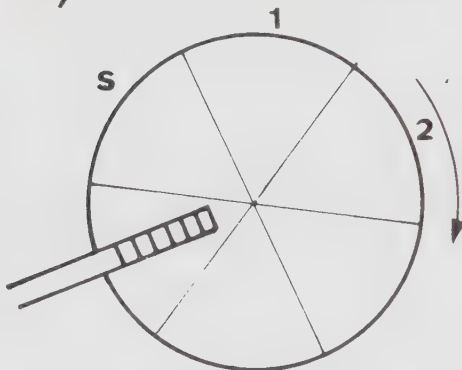
Lorsqu'un disque est placé sous le contrôle d'un système d'exploitation, celui-ci doit déterminer l'ordre dans lequel les demandes d'entrée/sortie seront satisfaites. La discipline de service la plus simple consiste à respecter l'ordre d'arrivée : premier arrivé - premier servi ("PAPS" ou "FIFO"). Supposant que les demandes apparaissent suivant un processus de Poisson, nous analyserons au paragraphe 2.1.3. un modèle à file d'attente du comportement du disque sous la discipline de service PAPS. Il est clair que celle-ci n'est pas optimale : supposons que le superviseur reçoive un ordre de transfert mettant en cause le secteur r ; avant que celui-ci se présente sous les têtes magnétiques, il se peut que ces têtes passent au-dessus d'un secteur s ($s \neq r$) attendu par un autre usager : cette seconde requête aurait pu être satisfaite sans qu'il en résulte de délai supplémentaire pour celle qui concerne le secteur r (voir figure 3a). Une politique plus efficace consiste à associer une file d'attente F_i à chaque secteur i : lorsque le secteur i s'engage sous les têtes, un transfert débute si la file F_i n'est pas vide ; F_i est généralement traitée selon le mode FIFO. C'est la politique d'Eschenbach [1] . Sur le plan pratique, l'implantation de cet algorithme n'est possible que s'il existe une instruction permettant au moniteur d'obtenir de l'unité de liaison le numéro du secteur présentement sous les têtes. Coffman [2] a développé un modèle de cet algorithme que nous présenterons au paragraphe 2.2.. La longueur de chaque file d'attente y est une variable aléatoire dont la distribution absolue est calculée ; aucune restriction n'est apportée à la longueur de ces files. Au contraire, Burge et Konheim [3] supposent que les demandes sont rangées dans une table pouvant en contenir au plus t . Supposons que cette table est continuellement pleine et qu'en outre la commutation des têtes est possible dans l'intervalle séparant deux secteurs consécutifs ; le nombre moyen de requêtes servies au cours d'une rotation est :



Légende: $(r, *)$ désigne
une demande d'E/S sur
le secteur r d'une piste
non précisée

Taux moyen λ

a) Politique PAPS



Taux moyen λ

b) Politique d'Eschenbach

Figure 3

$$\frac{2ts}{2t + s - 1}$$

où s désigne le nombre de secteurs du disque.

Dans les deux modèles ci-dessus, il est supposé que chaque transfert demandé est la lecture ou l'écriture d'un seul secteur de piste. Si les transferts sont de tailles différentes, leur ordonnancement optimal est un problème analogue à celui du "voyageur de commerce" [4].

1.2 DISQUES A TÊTES MOBILES.

Un tel dispositif se compose en fait d'une pile de n disques ($n \geq 1$). Lorsque n est strictement plus grand que 1, l'information est répartie par pistes sur les faces autres que les faces externes, supérieure et inférieure, qui, elles, servent de protection : il y a donc $2n - 2$ faces utilisées. Si n est égal à 1, les deux faces sont utilisées. Chaque piste est divisée en secteurs. Les pistes de numéro i de chacune des faces sont situées à la verticale les unes des autres et constituent le $i^{\text{ème}}$ cylindre.

Les piles, amovibles, sont vissées sur une "unité d'entraînement". Les têtes de lecture, au nombre d'une par face en général, sont portées par $(2n - 2)$ bras solidaires formant un "peigne" mobile (voir figure 4). La capacité d'une seule pile de disques est considérable ; elle atteint des dizaines de millions d'octets pour un coût peu élevé. La cadence de transfert est également importante. Mais le temps de réponse - défini comme l'intervalle séparant l'instant où la demande de lecture-écriture est émise par un usager et l'instant où le transfert s'achève - est moins bon que pour les disques à têtes fixes. Considérons en effet la manière dont est satisfaite une demande d'écriture ou de lecture sur le secteur s de la face f du cylindre c (voir figure 6) :

- S'il y a déjà d'autres demandes non satisfaites, celle que nous considérons doit attendre un temps W avant que le superviseur d'entrée/sortie ou le programme canal ne la transmette à l'unité de liaison.

- Les têtes de lecture ne se trouvant généralement pas au-dessus du cylindre recherché c lorsque le service débute, un mouvement du bras est nécessaire. Ce mouvement s'effectue en un temps T_1 qui est fonction du nombre n de cylindres à traverser.

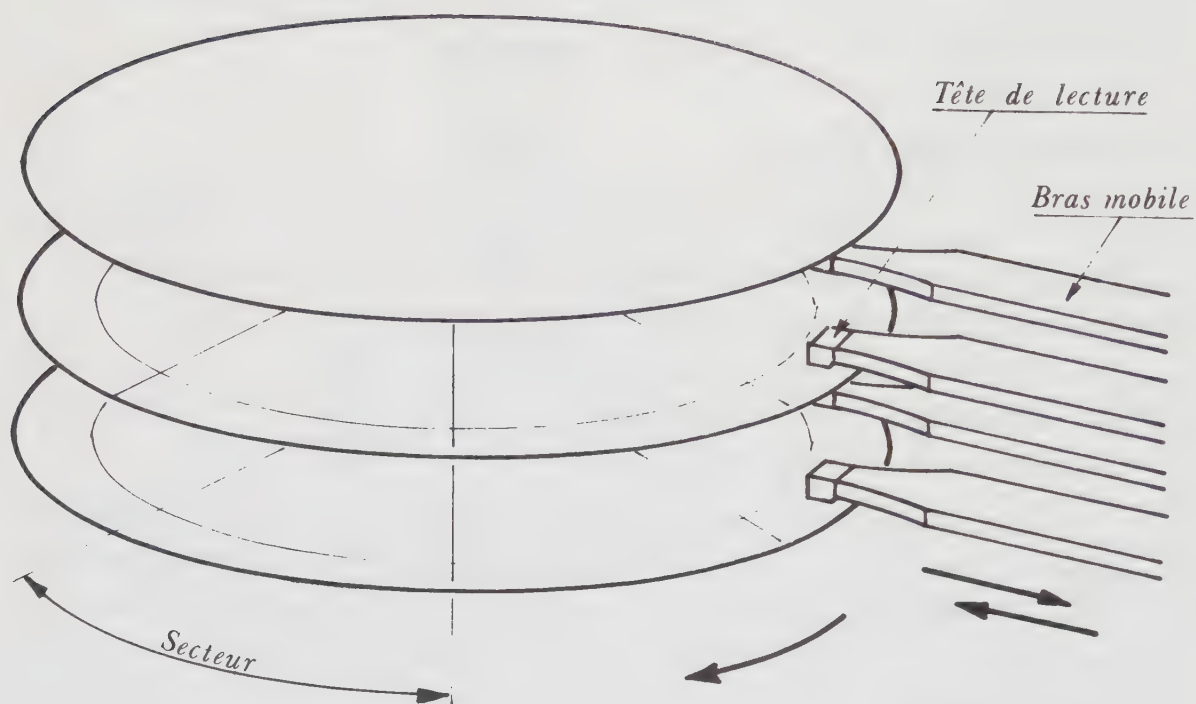


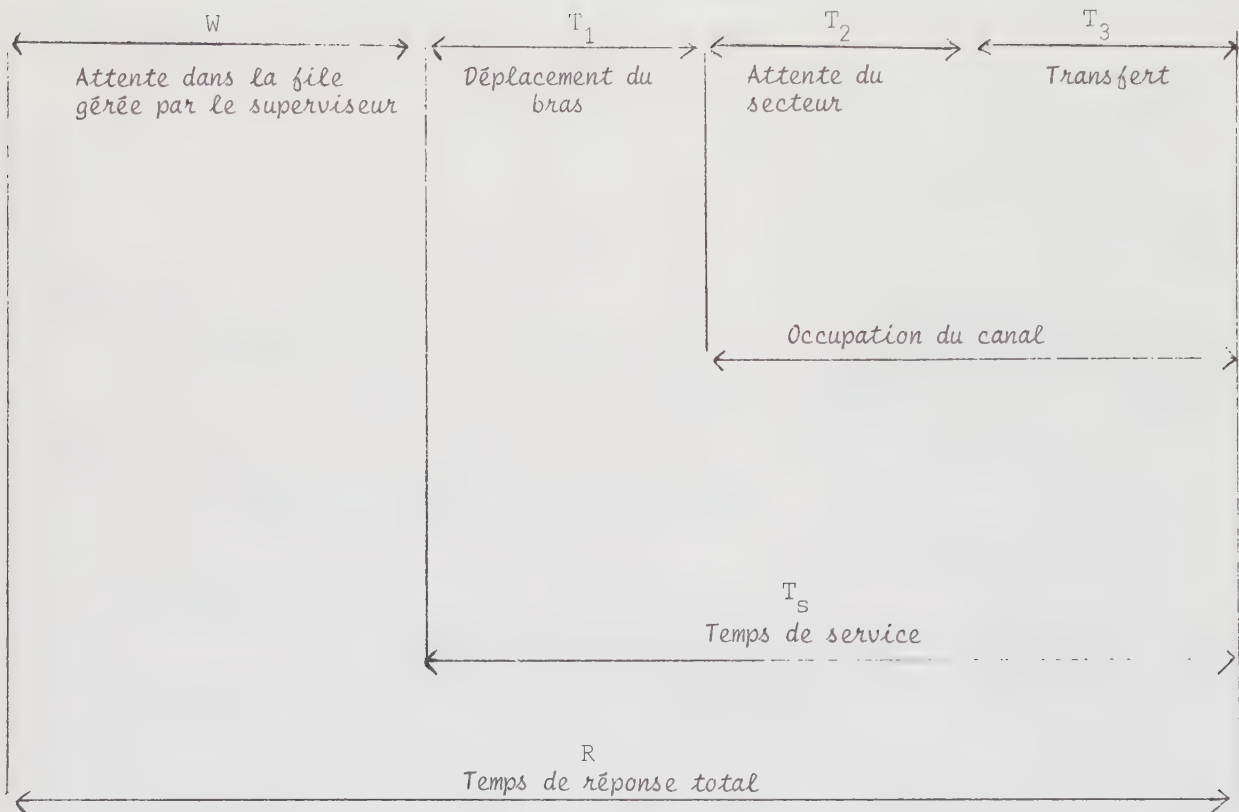
Figure 4

Vitesse de rotation	2 400 tours par minute
Durée d'une révolution	25,5 ms
Temps d'accès au cylindre	maxi 70 ms
	mini 10 ms
Nombre de disques par pile	11
Nombre de faces utilisées par pile	20
Nombre d'octets par secteur	1 024
Nombre de secteurs par piste	6
Nombre de pistes par cylindre	20
Nombre de cylindres par pile	400 (plus 6 en secours)
Nombre d'octets par pile	50 millions
Nombre de piles par unité de liaison	2 à 6

Caractéristiques des disques a têtes mobiles

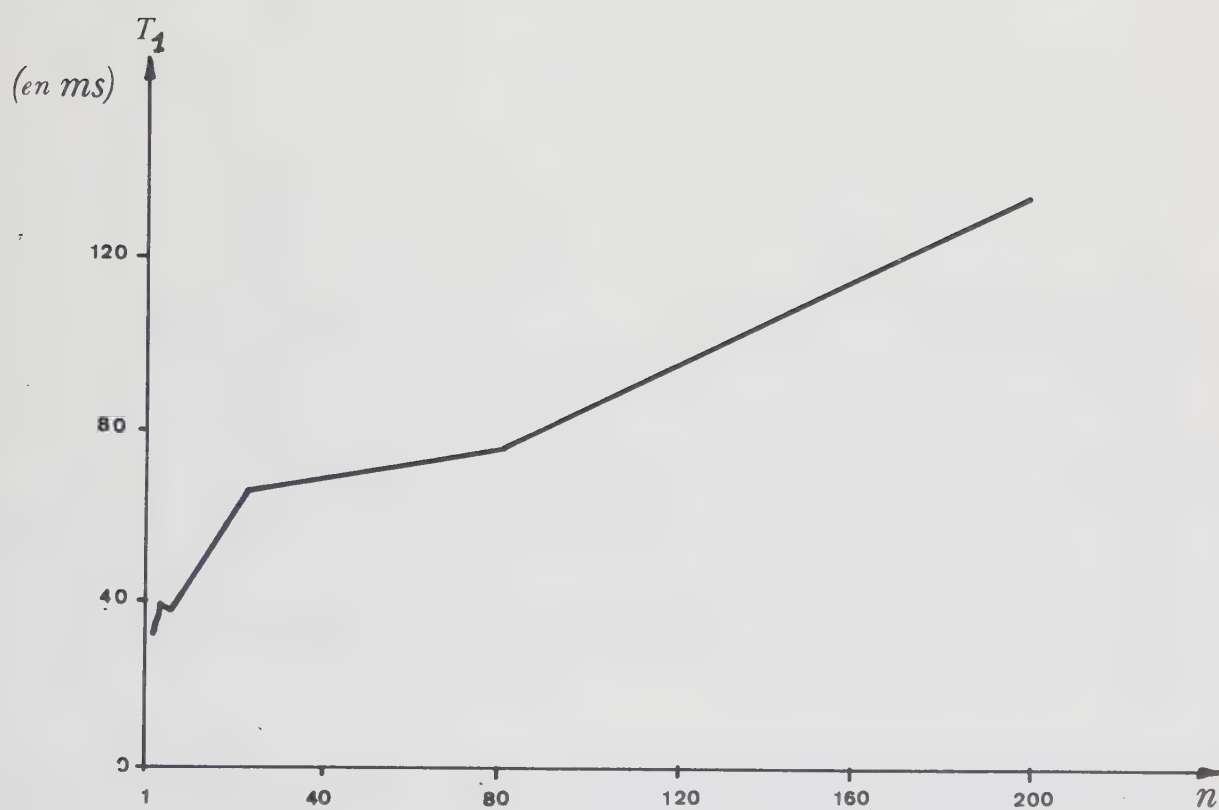
CII - MD50

Figure 5



Décomposition du temps de réponse d'un disque

Figure 6



Déplacement du bras (disques IBM 2314)

Figure 7

- Lorsque le bras est parvenu au-dessus du cylindre c , débute l'attente du passage du secteur s sous les têtes de lecture-écriture : le transfert débute sur la tête f au bout d'un temps T_2 .

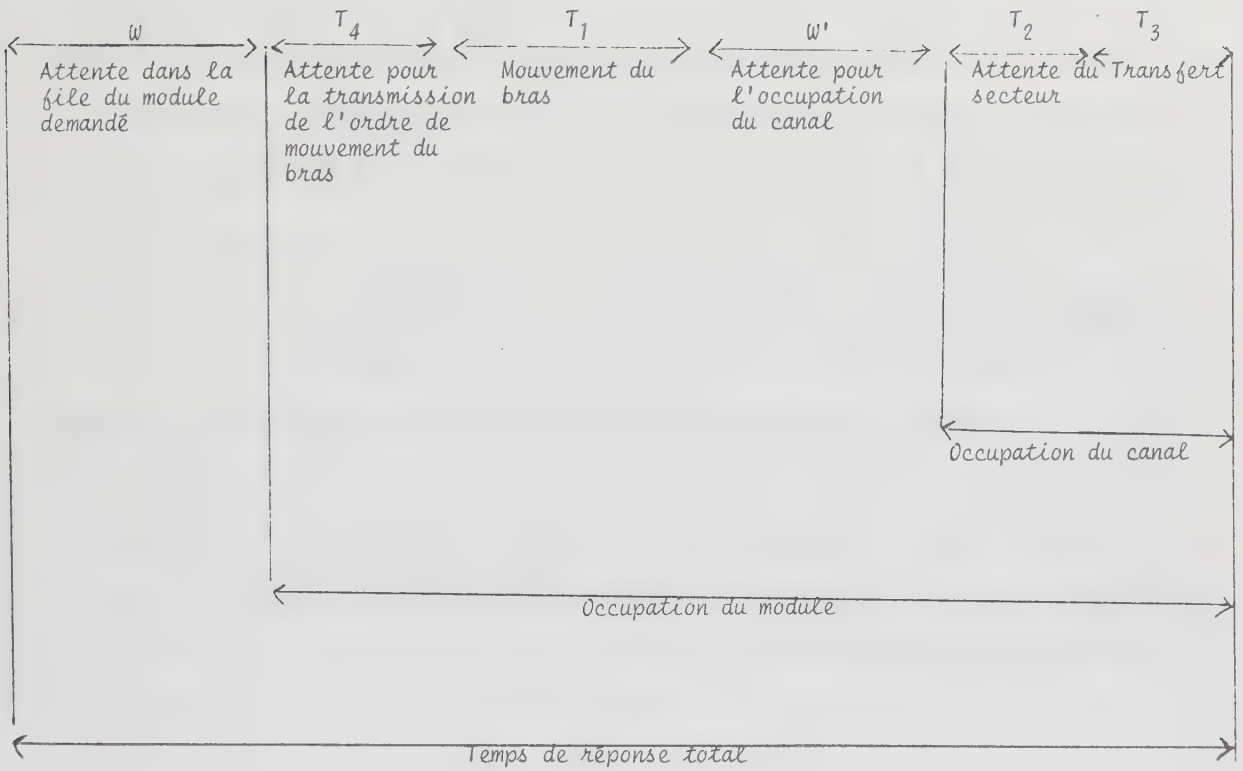
- Le transfert dure un temps T_3 à l'issue duquel la demande considérée est acquittée. Le cas échéant, une autre demande en attente peut commencer à être servie.

Le temps de transfert T_3 ne dépend que des caractéristiques techniques du disque et de la longueur de l'enregistrement à lire. En revanche une programmation adéquate de la file d'attente gérée par le moniteur permet de diminuer T_1 et T_2 . La figure 7 montre que T_1 croît assez rapidement avec le nombre n de cylindres que le bras doit traverser. En définissant des disciplines d'attente qui ordonnent les demandes non encore satisfaites en tenant compte des cylindres qu'elles référencent, il est possible de diminuer considérablement la valeur de n . Ces algorithmes ont été comparés en simulation par Teorey et Pinkerton [5] ; leur étude met également en évidence l'importance de l'optimisation de T_2 lorsque la fréquence des demandes est élevée. Le temps moyen de service sous diverses politiques a été calculé analytiquement dans [6] et [7]. Il est toutefois difficile de développer des modèles à file d'attente de ce type de disque. En effet les temps de service des diverses requêtes ne peuvent être considérés comme des variables aléatoires indépendantes ; la corrélation s'explique par le fait que la position des têtes de lecture à la fin du service d'une requête influe notablement sur le temps de service de la requête suivante.

1.3 LE PROBLEME DES GROUPES DE DISQUES.

Dans l'analyse du temps de réponse aux paragraphes précédents, nous avons négligé le fait que fréquemment plusieurs disques se partagent le même coupleur (ou unité de liaison). Cela est particulièrement vrai dans le cas des disques à têtes mobiles.

Appelons module l'ensemble formé par une pile de disques et le mécanisme d'entraînement et de transfert. Supposons que M modules utilisent le même canal. Une demande d'entrée-sortie pourra être mise en attente à deux niveaux. D'une part elle passera un temps W dans une file contrôlée par le superviseur si d'autres demandes concernant le même module ne sont pas satisfaites. D'autre part, au cours de son temps de service, elle pourra



Décomposition du temps de réponse d'un disque à têtes mobiles (multiplexage)

Figure 8

être retardée à cause de l'encombrement du canal.

La figure 8 représente le temps de réponse. On y retrouve les éléments W , T_1 , T_2 , T_3 composant le temps de réponse dans l'hypothèse du disque unique. En outre :

- en raison du multiplexage, l'ordre de mouvement du bras qui doit transiter par le coupleur des modules est retardé d'un temps T_4 si celui-ci est déjà occupé. Néanmoins cet ordre a priorité sur les transferts ; limité à l'attente du premier instant où la liaison coupleur-unité d'échange est interruptible, T_4 est très bref et peut être négligé vis à vis des autres termes composant le temps de réponse.

- lorsque le bras est positionné, la demande de transfert rejoint les demandes concernant d'autres modules qui se trouvent dans le même état ; elle n'occupera à son tour le canal qu'au bout d'un temps W .

S'appuyant en particulier sur des mesures en simulation, les auteurs de [8] traitent la file d'attente d'un module et la file du canal comme deux files indépendantes : cela signifie que les arrivées à la première mais aussi à la seconde seront supposées poissonniennes. En outre T_4 est négligé. Ces deux approximations permettent de trouver l'expression analytique du temps de réponse pour la discipline d'attente FIFO.

2 - L'UTILISATION DES DISQUES A TETES FIXES

Nous allons tout d'abord appliquer la technique de la chaîne de Markov incluse à l'étude de la file d'attente M/G/1 c'est-à-dire de la file d'attente à un seul serveur, à processus d'arrivée poissonnien et dont les temps de service sont des variables aléatoires positives indépendantes et identiquement distribuées. Ce type de file d'attente nous fournira un modèle pour la gestion Premier Arrivé-Premier Servi (PAPS) d'un disque à têtes fixes ainsi qu'une approximation simple du coût de la politique d'Eschenbach. Celle-ci sera ensuite analysée sur un modèle moins grossier. Les résultats obtenus seront comparés sur un exemple. Le dernier paragraphe sera consacré à l'utilisation d'un disque sans secteurs pour le rangement de segments.

2.1 ANALYSE DE LA FILE D'ATTENTE M/G/1.

Considérons une file d'attente à un seul serveur où les arrivées surviennent selon un processus de Poisson de taux moyen λ . Les temps de service sont des réalisations indépendantes d'une même variable aléatoire V dont la fonction de répartition sera notée B . On supposera que la variable aléatoire V a deux moments, c'est-à-dire que les intégrales $E(V) = \int_0^\infty v dB(v)$ et $E(V^2) = \int_0^\infty v^2 dB(v)$ convergent toutes deux. La discipline de service est : Premier Arrivé-Premier Servi (PAPS). Deux variables aléatoires nous intéressent particulièrement :

. $N(t)$ nombre des clients présents dans la file d'attente à l'instant t , que ces clients soient en attente ou en cours de service.

. R_n durée de l'intervalle entre l'arrivée du n^e client et la fin de son service : nous appellerons fréquemment cette quantité "temps de réponse du système".

2.1.1. La technique de la chaîne de Markov incluse.

Considérons un processus continu sur $\mathbb{R}^+$ c'est-à-dire une application qui à un réel positif t associe une variable aléatoire $X(t)$: un exemple en est le nombre de clients présents à l'instant t dans une file d'attente.

Soit t_0 un instant quelconque ; en général il n'est pas vrai que pour tout $t > t_0$

$$(1) \text{ distribution } [X(t) \mid X(t_0)] = \text{distribution } [X(t) \mid X(\tau) \text{ pour } \tau \leq t_0]$$

Si cette égalité est vérifiée pour tout $t > t_0$, on dit que t_0 est un point de régénération du processus. Un processus est markovien si et seulement si tout instant t est un point de régénération. S'il existe une suite croissante (t_q) ($q = 1, 2, 3, \dots$) de points de régénération, il est clair d'après l'égalité (1) que la suite de variables aléatoires obtenue en posant :

$$X_q = X(t_q)$$

est une chaîne de Markov. On dit que cette chaîne de Markov est incluse dans le processus X .

Considérons le nombre $N(t)$ de clients présents à l'instant t dans le système M/G/1 (ce nombre est défini de sorte que la fonction $N(\cdot)$ soit continue à droite en tout point, c'est-à-dire que sa valeur à l'instant d'un départ ou d'une arrivée est égale à sa valeur juste après cet instant.)

Cherchons les points de régénération de ce processus. Puisque la distribution des intervalles de temps entre deux arrivées a la propriété markovienne et comme la distribution des temps de service n'a pas cette propriété en général, les points de régénération τ doivent être tels que la durée de service obtenue avant τ par le client occupant le serveur à ce même instant ne puisse être utilisée pour déduire la durée de service restant à recevoir. En désignant par t_i l'instant du départ du $i^{\text{ème}}$ client, on définit

donc une suite croissante de points de régénération. Cette suite tend vers l'infini car la probabilité d'un nombre infini d'arrivées dans un temps fini - et donc celle d'un nombre infini de départs - est nulle.

Désignons par N_q la variable aléatoire représentant le nombre de clients dans le système juste après le $q^{\text{ème}}$ départ. Calculons les probabilités de transition :

$$p_{ij} = \Pr [N_{q+1} = j \mid N_q = i]$$

de cette chaîne de Markov. Puisque la distribution des temps de service et le processus d'arrivée sont tous deux stationnaires, p_{ij} est indépendant de q .

Soit g_k la probabilité de k arrivées pendant la durée d'un service. Le processus d'arrivée étant poissonnien, la probabilité de k arrivées dans le temps v est égale à $e^{-\lambda v} \frac{(\lambda v)^k}{k!}$ de sorte que :

$$(2) \quad g_k = \int_0^\infty e^{-\lambda v} \frac{(\lambda v)^k}{k!} dB(v)$$

g_k peut donc être déterminé analytiquement ou numériquement une fois que B est connue. Il n'est pas difficile de voir que pour $i \geq 1$:

$$p_{ij} = \begin{cases} g_{j-i+1} & \text{si } j \geq i-1 \\ 0 & \text{sinon} \end{cases}$$

Le cas $i = 0$ est légèrement différent. Si le système était vide après le $q^{\text{ème}}$ départ, pour qu'un départ puisse se produire à l'instant t_{q+1} , il est nécessaire qu'une arrivée ait eu lieu entretemps. Donc :

$$p_{0j} = g_j$$

Les probabilités de transition peuvent s'écrire sous forme d'une matrice stochastique infinie P :

$$P = \begin{array}{c|cccccc} \lambda & 0 & 1 & 2 & 3 & . & . \\ \hline 0 & g_0 & g_1 & g_2 & g_3 & . & . \\ 1 & g_0 & g_1 & g_2 & g_3 & . & . \\ 2 & 0 & g_0 & g_1 & g_2 & . & . \\ 3 & 0 & 0 & g_0 & g_1 & . & . \\ . & . & . & . & . & . & . \end{array}$$

Les coefficients g_i étant tous différents de 0, le graphe des transitions est fortement connexe et réflexif ; cela implique que la chaîne de Markov est irréductible et apériodique. Nous allons montrer qu'il existe un vecteur ligne (infini) unique $\pi = (\pi_0, \pi_1, \pi_2, \dots)$ tel que les deux égalités suivantes soient vérifiées :

$$\begin{aligned} \cdot \pi &= \pi P \\ \cdot \sum_{i=0}^{\infty} \pi_i &= 1 \end{aligned}$$

Ceci établira donc l'ergodicité de la chaîne de Markov. La composante π_i représente la limite lorsque q tend vers l'infini de la probabilité que le système soit dans l'état i après le $q^{\text{ème}}$ départ. C'est aussi la probabilité que le système en équilibre statistique soit dans l'état i . Nous allons déterminer la suite (π_i) par sa fonction génératrice :

$$\Pi(z) = \sum_{i=0}^{\infty} \pi_i z^i \quad (z \in [-1, 1])$$

L'égalité $\pi = \pi P$ se développe en :

$$(3) \quad \forall i \in \mathbb{N} : \pi_i = \pi_0 g_i + \sum_{k=1}^{i+1} \pi_k g_{i+1-k}$$

On reconnaît dans la sommation du membre de droite le $(i+1)^{\text{ème}}$ coefficient du développement en série entière de $\Pi(z) G(z)$ (où $G(\cdot)$ désigne la fonction génératrice de la suite (g_i)). En multipliant par z^i les deux membres de

l'équation (3) et en sommant membre à membre toutes les égalités ainsi obtenues, nous parvenons à :

$$\Pi(z) = \pi_0 G(z) + \frac{1}{z} (\Pi(z) - \pi_0) G(z)$$

C'est-à-dire :

$$\Pi(z) = \frac{\pi_0 (z-1) G(z)}{z-G(z)} \quad (z \neq 1)$$

La condition $\sum_{i=0}^{\infty} \pi_i = 1$ qui est équivalente à $\Pi(1) = 1$ nous permet de déterminer π_0 . En effet :

$$\Pi(1) = \lim_{z \rightarrow 1} \frac{\pi_0 G(z)}{\frac{z-G(z)}{z-1}} = \frac{\pi_0 G(1)}{(z-G(s))' \big|_{z=1}}$$

Donc :

$$(4) \quad \pi_0 = \frac{1-G'(1)}{G(1)} = 1 - G'(1)$$

De l'égalité (2) il résulte :

$$(5) \quad G(z) = \sum_{i=0}^{\infty} z^i \int_0^{\infty} e^{-\lambda v} \frac{(\lambda v)^i}{i!} dB(v) = \int_0^{\infty} e^{\lambda v(z-1)} dB(v)$$

La dérivation de cette égalité conduit à :

$$G'(z) = \lambda \int_0^{\infty} v e^{v(z-1)} dB(v)$$

et

$$G'(1) = \lambda \int_0^{\infty} v dB(v) = \lambda E(V)$$

La chaîne de Markov incluse sera donc ergodique si $\lambda E(V) < 1$, la distribution des probabilités à l'équilibre étant telle que

$$(6) \quad \Pi(z) = (1 - \lambda E(V)) \frac{(z-1) G(z)}{z - G(z)} \quad (z \neq 1)$$

2.1.2. Nombre d'utilisateurs dans le système M/G/1 à l'équilibre

La distribution qui est l'objet de la formule (6) est celle du nombre de clients dans le système en équilibre, ce nombre étant évalué juste après un départ. Soit $p_i(t)$ la probabilité qu'il y ait i clients dans le système à l'instant t . A priori π_i est une approximation de p_i , limite de

$p_i(t)$ lorsque t tend vers l'infini : c'est ce type de résultat que l'on peut généralement escompter de l'utilisation d'une chaîne de Markov incluse. Dans l'application particulière que nous traitons, p_i et π_i sont égaux. En utilisant des résultats généraux sur les processus récurrents, on peut montrer [9] que si $\lambda E(V) < 1$, le nombre d'utilisateurs dans la file M/G/1 a une distribution limite telle que $\pi_0 = p_0$. Nous allons déduire de ce théorème la valeur de p_n en fonction de $\pi_0, \pi_1, \dots, \pi_n$. Soit τ un instant quelconque où l'on observe le système en équilibre statistique. Supposons qu'à l'instant τ le serveur soit occupé par un usager $\tilde{C}$ dont le service S de durée $\tilde{V}$ a débuté à l'instant τ_1 et s'achève à l'instant τ_2 :

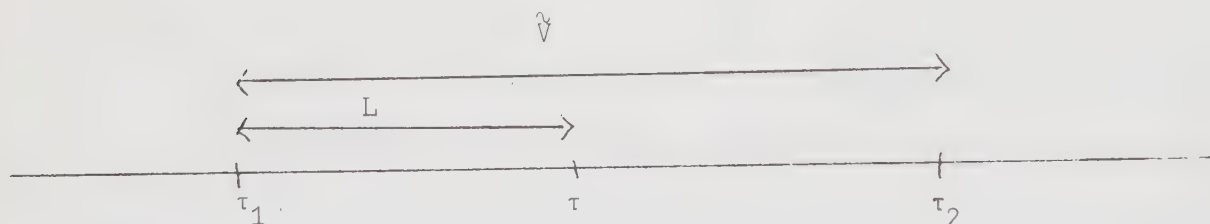


Figure 9

La distribution de la variable aléatoire $L = \tau - \tau_1$ peut être définie par :

$$\Pr [L \geq x] = \int_x^\infty \Pr [A_1] \Pr [A_2]$$

où l'on a posé :

$$A_1 = [\bar{y} < \tilde{V} < y + dy \mid \text{le serveur est actif en } \tau]$$

et

$$A_2 = \left[\begin{array}{l} \text{l'instant d'observation est à moins de} \\ (y-x) \text{ unités de temps de la fin de } S \end{array} \middle| A_1 \right]$$

$$= \left[|\tau_2 - \tau_1| < y - x \middle| A_1 \right]$$

Sur une très longue période, la proportion du temps d'activité du serveur dûe aux services dont la durée est comprise entre y et $y + dy$ est égale à $\frac{1}{E(V)} y dB(y)$ au second ordre près. Cette quantité est la probabilité de l'évènement A_1 . Par ailleurs il est clair que $\Pr [A_2] = \frac{y-x}{y}$. Par suite :

$$\Pr [\bar{L} \geq x] = \frac{1}{E(V)} \int_x^\infty (y-x) dB(y)$$

ce qui entraîne, au second ordre près :

$$\Pr [x \leq L < x + dx] = \frac{dx}{E(V)} \int_x^\infty dB(y) = \frac{1-B(x)}{E(V)} dx$$

La probabilité que k usagers se soient joints à la file d'attente dans l'intervalle $[\tau_1, \tau]$ est donc égale à :

$$(10) \quad h_k = \frac{1}{E(V)} \int_0^\infty \frac{e^{-\lambda x} (\lambda x)^k}{k!} (1-B(x)) dx$$

Le fait qu'à l'instant τ_1 de son début de service l'utilisateur C est seul dans le système est équivalent au fait que l'utilisateur précédent est parti en laissant 0 ou 1 usager en attente : la probabilité de cet évènement est donc $\pi_0 + \pi_1$. Pour $n \geq 1$, la probabilité de la présence de n usagers (y compris U) à l'instant τ_1 est, par définition, π_n . Donc, pour tout entier strictement positif n :

$$p_n = \Pr [\text{le serveur est actif à l'instant } \tau] \times$$

$$\sum_{i=1}^n \Pr [i \text{ usagers présents à } \tau_1] \times \Pr [(n-i) \text{ arrivées en } \tau_1 \text{ et } \tau]$$

$$= \lambda E(V) \times (\pi_n h_0 + \pi_{n-1} h_1 + \dots + \pi_2 h_{n-2} + (\pi_1 + \pi_0) h_{n-1})$$

ou, plus simplement,

$$(11) \quad p_n = \lambda \int_0^\infty u_{n-1}(x) (1 - B(x)) dx$$

où nous avons posé :

$$u_{n-1}(x) = \pi_n e^{-\lambda x} + \pi_{n-1} \frac{\lambda x e^{-\lambda x}}{1!} + \dots + \pi_2 \frac{(\lambda x)^{n-2} e^{-\lambda x}}{(n-2)!} +$$

$$(\pi_1 + \pi_0) \frac{(\lambda x)^{n-1} e^{-\lambda x}}{(n-1)!}$$

Remarquons la suite de fonctions u_n est solution du système différentiel

$$(12) \quad u'_n = \lambda(u_{n-1} - u_n) \quad (n = 1, 2, \dots)$$

L'égalité (3) (voir § 2.1.1.) est équivalente à :

$$(13) \quad \pi_n = \int_0^\infty u_n(x) dB(x) \quad (n = 1, 2, \dots)$$

Faisons une intégration par parties :

$$\begin{aligned} \pi_n &= u_n(0) + \int_0^\infty u'_n(x) (1 - B(x)) dx \\ &= \pi_{n+1} + \lambda \int_0^\infty (u_{n-1}(x) - u_n(x)) (1 - B(x)) dx \\ &= \pi_{n+1} + p_n - p_{n+1} \end{aligned}$$

Il en résulte l'égalité $\pi_n - p_n = \pi_{n+1} - p_{n+1}$ qui est vérifiée pour tout $n > 1$. Comme les séries de terme général π_n et p_n sont convergentes, il en est de même de leur différence ce qui implique donc :

$$\forall n \in \{1, 2, \dots\} \quad \pi_n = p_n$$

π_0 et p_0 étant aussi égaux, nous pouvons énoncer le théorème suivant :

Théorème 1 :

Soit un système M/G/1 dont le taux moyen d'arrivées est λ et dont la durée de service est une variable aléatoire positive V , de fonction de répartition B . Supposons que V ait une moyenne $E(V)$. Si $\rho = \lambda E(V)$ est strictement inférieur à 1, la distribution du nombre de clients dans le système tend vers une distribution d'équilibre dont la fonction génératrice est :

$$P(z) = (1 - \rho) \frac{(1-z) G(z)}{G(z) - z} \quad (z \in [-1, 1[)$$

où
$$G(z) = \int_0^\infty e^{\lambda v(z-1)} dB(v)$$

Théorème 2 (Formule de Pollaczek et Khintchine)

Si les deux premiers moments de V existent et si $\rho < 1$, le nombre moyen d'utilisateurs dans le système est, à l'équilibre :

$$(14) \quad \bar{N} = \rho + \frac{\lambda^2 E(V^2)}{2(1-\rho)}$$

et le temps de réponse moyen est :

$$(15) \quad \bar{R} = E(V) + \frac{\lambda E(V^2)}{2(1-\rho)}$$

L'égalité (14) est une conséquence immédiate du théorème 1 puisque $\bar{N} = \lim_{z \rightarrow 1} P'(z)$

Démontrons l'égalité (15).

Soit R le temps que passe dans le système un utilisateur C^* , C^* a la probabilité p_0 d'arriver alors que le serveur est inactif : dans ce cas, son séjour dans le système se limite au temps qui lui est nécessaire pour être servi.

$$(16) \quad E(h \mid \text{le serveur est inactif à l'arrivée de } C^*) = E(V)$$

Avec la probabilité $(1 - p_0)$, C^* arrive alors qu'un autre utilisateur, $\tilde{C}$, requérant un service de durée $\tilde{V}$ occupe le serveur. Nous désignerons par L le temps de service déjà obtenu par $\tilde{C}$ à l'instant de l'arrivée de C^* et par M le temps de service restant à obtenir.

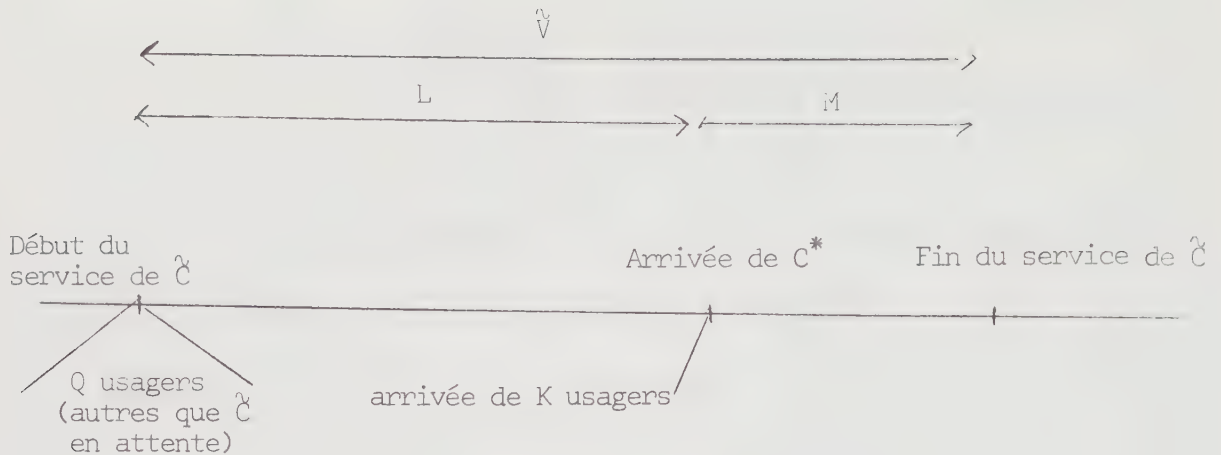


Figure 10

Comme nous l'avons déjà remarqué :

$$(17) \quad \Pr [y \leq \hat{V} < y + dy] = \frac{y \, dB(y)}{E(V)} + o(dy)$$

En outre, puisque l'instant de l'arrivée de C^* est uniformément distribué entre le début et la fin du service de C :

$$(18) \quad \Pr [L \leq x \mid y \leq \hat{V} < y + dy] = \frac{x}{y} + o(dy) \quad (0 \leq x < y)$$

Si le serveur est actif lors de l'arrivée de C^* , le temps R que cet usager passe dans le système est la somme de quatres termes :

a) M , temps de service résiduel de l'usager C

$$\begin{aligned} E(M \mid y \leq \hat{V} < Y + dy) &= \int_0^{\hat{V}} (\hat{V} - x) \frac{dx}{\hat{V}} \\ &= \int_0^y (y - x) \frac{dx}{y} + o(dy) \\ &= \frac{y}{2} + o(dy) \end{aligned}$$

b) Le temps pour servir les Q autres usagers présents à l'instant où le service de C a débuté. Comme nous l'avons remarqué au cours de la

démonstration de l'égalité des suites (π_n) et (p_n) , la distribution de Q est telle que :

$$\Pr [Q = n] = \begin{cases} \pi_{n+1} & \text{si } n \geq 1 \\ \pi_0 + \pi_1 & \text{si } n = 0 \end{cases}$$

Cette contribution au temps de réponse R est évidemment indépendante de la valeur de $\hat{V}$; elle a pour espérance mathématique :

$$\begin{aligned} E(V) \times E(Q) &= E(V) \times \sum_{n=1}^{\infty} n \pi_{n+1} \\ &= E(V) \left[\sum_{n=0}^{\infty} (n+1) \pi_{n+1} - \sum_{n=0}^{\infty} \pi_{n+1} \right] \\ &= E(V) [\bar{N} - \rho] \end{aligned}$$

c) Le temps nécessaire pour servir les K usagers qui se sont joints au système entre le début du service de $\hat{C}$ et l'arrivée de C^* . La distribution de K dépend de L, donc de $\hat{V}$. La valeur moyenne de cette contribution à R conditionnée par :

$$y \leq \hat{V} < y + dy$$

est d'après (18) :

$$\begin{aligned} E(V) E(K \mid y \leq \hat{V} < y + dy) &= E(V) \lambda E(L \mid y \leq \hat{V} < y + dy) \\ &= \lambda E(V) \left[\int_0^y \frac{x}{y} dx + o(dy) \right] \\ &= \frac{\rho y}{2} + o(dy) \end{aligned}$$

d) Le temps de service de C^* lui-même.

Par suite

$$E(R \mid y \leq \hat{V} < y + dy) = E(V) [\bar{N} + (1 - \rho)] + \frac{y}{2} (1 + \rho) + o(dy)$$

Utilisons l'égalité (17)

$E(R \mid \text{le serveur est actif à l'arrivée de } C^*) =$

$$\begin{aligned} & \int_0^\infty E(R \mid y \leq \tilde{V} < y + dy) \frac{y \, dB(y)}{E(V)} \\ (19) \quad & = \frac{E(V^2)}{2E(V)} (1 + \rho) + E(V) [\bar{N} + 1 - \rho] \end{aligned}$$

Rapprochons les égalités (16) et (19) :

$$\begin{aligned} \bar{R} &= p_0 E(R \mid \text{serveur non actif}) + (1 - p_0) E(R \mid \text{serveur actif}) \\ &= (1 - \rho) E(R \mid \text{serveur non actif}) + \rho E(R \mid \text{serveur actif}) \\ &= E(V) + \rho E(V) [\bar{N} - \rho] + \frac{E(V^2)}{2E(V)} \rho(1 + \rho) \end{aligned}$$

Reportons dans ce résultat la valeur déjà trouvée pour N :

$$\bar{R} = E(V) + E(V^2) \left[\frac{\rho E(V) \lambda^2}{2(1 - \rho)} + \frac{(1 + \rho) \rho}{2E(V)} \right]$$

soit, puisque $\rho = \lambda E(V)$

$$\begin{aligned} \bar{R} &= E(V) + \frac{E(V^2)}{2} \lambda \left[\frac{\rho^2}{1 - \rho} + 1 + \rho \right] \\ &= E(V) + \frac{\lambda E(V^2)}{2} \frac{1}{1 - \rho} \end{aligned}$$

ce qui est le résultat annoncé.

Corollaire :

A l'équilibre le nombre moyen d'utilisateurs présents dans le système M/G/1 et le temps moyen de réponse sont liés par la relation

$$(20) \quad \lambda \bar{R} = \bar{N}$$

Ce corollaire se déduit immédiatement de la confrontation des égalités (14) et (15).

L'égalité $\lambda \bar{R} = \bar{N}$ est vérifiée dans une très large classe de systèmes d'attente : elle l'est en particulier dans les systèmes à un serveur où les arrivées constituent un processus récurrent et où les temps de service sont des variables aléatoires indépendantes identiquement distribuées (système GI/G/1), quel que soit par ailleurs l'ordre dans lequel les utilisateurs en attente

accèdent au serveur. Ce résultat fondamental est connu sous le nom de théorème de Little ([10] , [11]).

2.1.3. Premières applications de la formule de Pollaczek et Khintchine

Considérons un disque à têtes fixes dont la période de rotation est T. Nous faisons les hypothèses suivantes sur son fonctionnement :

H₁) la surface du disque est divisée en S secteurs égaux ,

H₂) L'arrivée au superviseur des demandes de transfert est un processus de Poissons de taux λ . La taille des enregistrements à transmettre est la capacité d'un secteur de piste.

H₃) La probabilité qu'une requête concerne un secteur déterminé est égale à 1/S. Il n'y a pas de corrélation entre les numéros des secteurs faisant l'objet des demandes de transfert successives.

Sous ces hypothèses la formule de Pollaczek et Khintchine nous permet de calculer le nombre moyen de demandes encore non satisfaites lorsque la politique de service est : Premier Arrivé - Premier Servi. Le temps de service V est la somme du temps de transfert T/S et du temps d'accès qui a la valeur kT/S (k = 0, 1,...S-1) avec probabilité 1/S. Il s'agit en fait d'une approximation puisque le temps d'accès est réparti entre 0 et T pour une demande qui survient alors que le système est vide. On peut toutefois montrer que cette approximation est bonne. Elle sera d'ailleurs d'autant meilleure que le cas évoqué ci-dessus est plus rare (p_0 petit) ou que S est plus grand. Dans ce modèle V est une variable aléatoire qui vaut kT/S (k = 1, 2,...S) avec la probabilité 1/S. Ses deux premiers moments sont donc :

$$E(V) = \frac{S+1}{2S} T$$

et

$$E(V^2) = \frac{(S+1)(2S+1)}{6S^2} T^2$$

Lorsque le système a atteint son équilibre, le nombre moyen de demandes momentanément insatisfaites (y compris éventuellement celle en cours de traitement) est donc :

$$(21) \quad \bar{N}_{\text{PAPS}} = \lambda T \frac{S+1}{2S} + \lambda^2 T^2 \frac{(S+1)(2S+1)}{12S^2(1 - \frac{S+1}{2S} \lambda T)}$$

et le temps moyen de réponse est :

$$(22) \quad \bar{R}_{\text{PAPS}} = \frac{1}{\lambda} \bar{N}_{\text{PAPS}}$$

La formule de Pollaczek et Khintchine nous fournit également une valeur approchée $\bar{N}_{\text{EA}}$ du nombre moyen de demandes insatisfaites lorsque le disque est géré selon la politique d'Eschenbach. Faisons les mêmes hypothèses que ci-dessus sur le disque et sur son environnement. En particulier, il résulte de H_2 et H_3 que le processus d'arrivée dans la file F_r (c'est-à-dire la file attachée au secteur r , $r = 1, 2, \dots, S$) est poissonnien de taux $\frac{\lambda}{S}$. Soit $N_r(t)$ le nombre de requêtes dans cette file - y compris éventuellement celle en cours de traitement - à l'instant t . Les variables aléatoires $N_r(t)$ ($r = 1, 2, \dots, S$) ne sont pas stochastiquement indépendantes. Néanmoins, par symétrie, elles ont toutes la même distribution absolue. Nous étudierons donc l'une d'elles en omettant de préciser l'indice r afin d'alléger les notations.

Si le facteur d'utilisation du disque était suffisamment important pour que la file d'attente considérée ne soit jamais vide, l'intervalle entre les instants d'achèvement de deux transferts consécutifs serait T : nous approximerons la distribution de $N(t)$ par celle du nombre d'utilisateurs dans une file M/G/1 dont le taux d'arrivée est $\frac{\lambda}{S}$ et dont le temps de service V a la valeur T avec probabilité 1.

La valeur moyenne du nombre de clients dans une des files d'attente de l'algorithme d'Eschenbach a donc pour valeur approchée :

$$(23) \quad \bar{N}_{EA} = \rho + \frac{\rho^2}{2(1-\rho)}$$

$$\text{où } \rho = \frac{\lambda T}{S}$$

Dans cette approximation le nombre total de demandes insatisfaites est donc $S \times \bar{N}_{EA}$ et le temps de réponse moyen est $\frac{\lambda}{S} \bar{N}_{EA}$.

Nous allons maintenant présenter un modèle plus fin de l'algorithme d'Eschenbach qui nous permettra d'apprécier la valeur de la présente approximation.

2.2 ANALYSE DE LA POLITIQUE D'ESCHENBACH. [2]

2.2.1. Détermination du nombre de demandes en attente

Considérons à nouveau un disque satisfaisant aux hypothèses H_1 , H_2 et H_3 . Nous nous proposons d'étudier le nombre N de demandes de transfert encore insatisfaites figurant dans une file d'attente particulière F , lorsque ce disque est géré selon la politique d'Eschenbach.

Choisissons pour origine des temps un instant où la fin du secteur considéré passe sous les têtes de lecture : nous supposons que la file d'attente est alors vide. Les départs ne pourront avoir lieu qu'aux instants $T, 2T, 3T, \dots$. Nous utiliserons les notations suivantes :

. $N(t)$ est le nombre de demandes de transferts insatisfaites à l'instant t . En particulier $N(qT) = N(qT + 0)$ ($q \in \mathbb{N}$)

$$. p(i, t) = \Pr [N(t) = i \mid N(0) = 0] \quad (i \in \mathbb{N}, t \in \mathbb{R}^+)$$

$$. \pi_i = \lim_{q \rightarrow \infty} p(i, qT)$$

$$. P(z, t) = \sum_{i=0}^{\infty} p(i, t) z^i \quad (z \in [-1, 1])$$

$$. \Pi(z) = \sum_{i=0}^{\infty} \pi_i z^i \quad (z \in [-1, 1])$$

. $A_\tau(z) = e^{\frac{-\lambda\tau}{S}(1-z)}$: c'est la fonction génératrice de la distribution du nombre des arrivées selon un processus de Poisson de taux $\frac{\lambda}{S}$ au cours d'un intervalle de temps de durée τ .

On vérifie sans peine que la suite de variables aléatoires $(N(qT))_{q \in \mathbb{N}}$ constitue une chaîne de Markov que nous noterons $\mathcal{C}$.

Théorème

Si $\lambda T < S$, la chaîne de Markov $\mathcal{C}$ est ergodique. Sa distribution d'équilibre a pour fonction génératrice :

$$(24) \quad \Pi(z) = \left(1 - \frac{\lambda T}{S}\right) \frac{(1-z) e^{-\frac{\lambda T}{S^2}(1-z)}}{e^{-\frac{\lambda T}{S}(1-z)} - z} \quad (z \neq 1)$$

Soient p_{ij} ($i \in \mathbb{N}$, $j \in \mathbb{N}$) les probabilités de transition de la chaîne de Markov $\{ \}$ et soit a_n (resp. a'_n , a''_n) la probabilité de n arrivées selon un processus de Poisson de taux λ/S au cours d'un intervalle de temps de durée T (resp. $\frac{S-1}{S} T$, $\frac{T}{S}$); la fonction génératrice de la distribution (a_n) (resp. (a'_n) , (a''_n)) est donc A_T (resp. $A_{\frac{S-1}{S} T}$, $A_{\frac{T}{S}}$).

Calculons les probabilités de transition. Si $N(qT)$ n'est pas nul, une requête sera satisfaite entre $(q + \frac{S-1}{S}) T$ et $(q+1) T$. Donc, pour tout i strictement positif :

$$(25) \quad p_{ij} = \begin{cases} 0 & \text{si } j < i-1 \\ a_{j-i+1} & \text{sinon} \end{cases}$$

Le calcul de p_{0j} est un peu plus délicat car il n'y aura de transfert effectué au cours de la rotation qu'à condition qu'il en arrive une demande suffisamment tôt : plus précisément, si la file d'attente est vide à l'instant qT il faudra une arrivée dans l'intervalle $I_1 =]qT, (q + \frac{S-1}{S}) T[$, les demandes parvenant pendant l'intervalle $I_2 = [(q + \frac{S-1}{S}) T, (q+1) T]$ ne pouvant être traitées au plus tôt qu'au cours de la rotation suivante. Donc :

$$\begin{aligned} p_{0j} &= \Pr [N((q+1) T) = j \mid N(qT) = 0] \\ p_{0j} &= \sum_{r=1}^{j+1} \Pr [r \text{ arrivées dans } I_1 \text{ et } (j+1-r) \text{ arrivées dans } I_2] \\ &\quad + \Pr [0 \text{ arrivée dans } I_1 \text{ et } j \text{ arrivées dans } I_2] \\ p_{0j} &= \sum_{r=1}^{j+1} a'_r a''_{j+1-r} + a'_0 a''_j \end{aligned}$$

$$(26) \quad P_{0j} = \sum_{r=0}^{j+1} a'_r a''_{j+1-r} - a'_0 a''_{j+1} + a'_0 a''_j$$

Il résulte de (25) et (26) que la chaîne de Markov $\mathcal{E}$ est irréductible et apériodique. Pour s'assurer de son ergodicité, il suffit donc de vérifier qu'il existe une suite (π_ℓ) de réels positifs tels que :

$$(27) \quad \begin{aligned} & \cdot \sum_{\ell=0}^{\infty} \pi_\ell = 1 \\ & \cdot \pi_\ell = \sum_{i=1}^{\ell+1} \pi_i a_{\ell-i+1} + \pi_0 \sum_{r=0}^{\ell+1} a'_r a''_{\ell+1-r} \\ & \quad - \pi_0 a'_0 a''_{\ell+1} + \pi_0 a'_0 a''_\ell \quad (\ell = 0, 1, 2, \dots) \end{aligned}$$

Multiplions les deux membres de (21) par $z^{\ell+1}$ et sommons toutes les égalités ainsi obtenues.

$$(28) \quad \begin{aligned} z \Pi(z) &= \sum_{\ell=0}^{\infty} \sum_{i=0}^{\ell+1} \pi_i a_{\ell-i+1} z^{\ell+1} - \pi_0 \sum_{\ell=0}^{\infty} a_{\ell+1} z^{\ell+1} \\ &+ \pi_0 \sum_{\ell=0}^{\infty} \sum_{r=0}^{\ell+1} a'_r a''_{\ell+1-r} z^{\ell+1} - \pi_0 a'_0 \sum_{\ell=0}^{\infty} a''_{\ell+1} z^{\ell+1} \\ &+ \pi_0 a'_0 z \sum_{\ell=0}^{\infty} a''_\ell z^\ell \\ &= (\Pi(z) A_T(z) - \pi_0 a_0) - \pi_0 (A_T(z) - a_0) \\ &+ \pi_0 \left(\frac{A_{S-1}}{S} T \right) \left(\frac{z}{S} A_T(z) - a'_0 a''_0 \right) - \pi_0 a'_0 \left(\frac{A_T(z)}{S} - a''_0 \right) \\ &+ \pi_0 a'_0 z \frac{A_T(z)}{S} \end{aligned}$$

Remarquons que $\frac{A_{S-1}}{S} T \times \frac{A_T}{S} = A_T$ et que $a'_0 a''_0 = a_0$; l'équation (28) se dérive donc en :

$$(29) \quad \Pi(z) \frac{(z - A_T(z))}{z-1} = \pi_0 a'_0 \frac{A_T(z)}{S} \quad (z \neq 1)$$

Pour tenir compte de la contrainte $\sum_{\ell=0}^{\infty} \pi_\ell = 1$, faisons tendre z vers 1 dans les deux membres de l'égalité ci-dessus, ce qui nous donnera la valeur de π_0 ,

dans le cas où l'équilibre existe :

$$(30) \quad (z - A_T(z))|_{z=1} = 1 - \frac{\lambda}{S} T = \pi_0 a'_0$$

La condition d'ergodicité est donc $\lambda T < S$; l'égalité (24) résulte immédiatement de (29) et (30).

De la distribution du nombre de demandes insatisfaites à l'instant où la fin du secteur passe sous les têtes de lecture-écriture, on déduit aisément la distribution à tout instant. Ceci fait l'objet du corollaire ci-dessous.

Corollaire 1

Si $\lambda T < S$, pour tout réel α appartenant à l'intervalle $[0, 1[$, $P(z, (\alpha+q)T)$ ($q \in \mathbb{N}$) tend vers :

$$(31) \quad P_\alpha(z) = \left(1 - \frac{\lambda T}{S}\right) \frac{e^{-\frac{\lambda T}{S^2}(1-z)} e^{-\frac{\lambda T \alpha}{S}(1-z)}}{e^{-\frac{\lambda T}{S}(1-z)} - z} \quad (z \neq 1)$$

lorsque q tend vers l'infini.

Comme aucun départ ne peut avoir lieu dans l'intervalle $]qT, (q+\alpha)T]$, la variable aléatoire $N((q+\alpha)T)$ est la somme de la variable aléatoire $N(qT)$ et du nombre d'arrivées pendant l'intervalle de durée αT . Or la fonction génératrice attachée à la somme de deux variables aléatoires indépendantes est le produit des fonctions génératrices de chacune d'elles. Par suite :

$$(32) \quad P(z, (\alpha+q)T) = P(z, qT) A_{\alpha T}(z)$$

égalité d'où l'on tire immédiatement le corollaire.

Définition

Le nombre moyen $\bar{N}_E$ de demandes insatisfaites dans la file F est la limite -si elle existe- de l'expression:

$$(33) \quad \frac{\int_0^\theta N(t) dt}{\theta} \quad (\theta \in \mathbb{R}^{+*})$$

lorsque θ tend vers l'infini.

Corollaire 2

Si $\lambda T < S$, N_E existe et est égal à :

$$(34) \quad \bar{N}_E = \rho \left(\frac{S+2}{2S} \right) + \frac{\rho^2}{2(1-\rho)}$$

où l'on a posé $\rho = \frac{\lambda T}{S}$.

Soit q le plus grand entier inférieur ou égal à $\frac{\theta}{T}$. L'expression

(33) est égale à :

$$(35) \quad \frac{qT}{\theta} \frac{1}{T} \left[\frac{\sum_{\ell=0}^{q-1} \int_{\ell T}^{(\ell+1)T} N(t) dt}{q} \right] + \frac{1}{\theta} \int_{qT}^{\theta} N(t) dt$$

La remarque qui nous a permis d'écrire l'égalité (32) permet également d'écrire :

$$(36) \quad \bar{N}(t) = \bar{N}(\ell T) + \frac{\lambda}{S} (t - \ell T)$$

pour tout t appartenant à $[\ell T, (\ell+1) T[$. Donc :

$$\int_{\ell T}^{(\ell+1)T} N(t) dt = T \bar{N}(\ell T) + \frac{\lambda}{2S} T^2$$

La suite de variables aléatoires $(N(\ell T))$ ($\ell = 0, 1, 2, \dots$) convergeant en loi vers la variable aléatoire dont la distribution a pour fonction génératrice Π , $\lim_{\ell \rightarrow \infty} N(\ell T)$ est égal à la moyenne de la variable aléatoire limite c'est-à-dire à $\lim_{z \rightarrow 1} \Pi'(z)$. Cette dernière limite peut être calculée à partir de

l'expression fournie par le théorème : elle vaut $\frac{\lambda T}{S^2} + \frac{\left(\frac{\lambda T}{S}\right)^2}{2(1 - \frac{\lambda T}{S})}$. La suite

de terme général $\int_{\ell T}^{(\ell+1)T} N(t) dt$ tend donc vers $T \left(\rho \frac{S+2}{2S} + \frac{\rho^2}{2(1-\rho)} \right)$ qui,

d'après un théorème bien connu, est aussi la limite de $\frac{1}{q} \sum_{\ell=0}^{q-1} \int_{\ell T}^{(\ell+1)T} N(t) dt$.

Par ailleurs il est aisé de montrer que $\int_{qT}^{\theta} N(t)dt$ possède un majorant indépendant de q et θ . Comme en outre $\frac{qT}{\theta}$ tend vers 1 lorsque θ tend vers l'infini, la limite de (35) est bien le nombre annoncé.

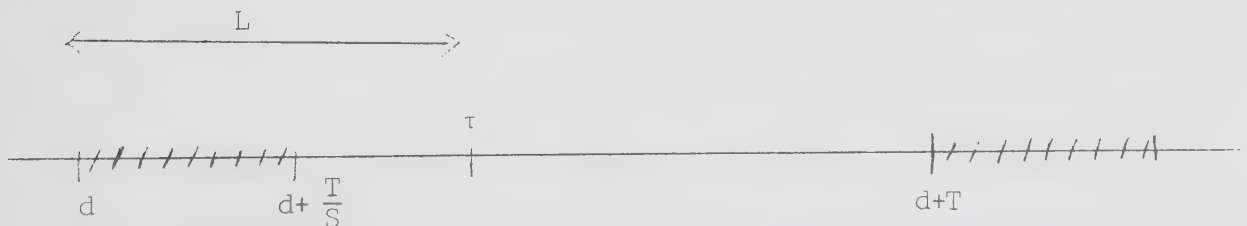
Corollaire 3 :

A l'équilibre, le temps de réponse moyen d'un disque géré selon la politique d'Eschenbach est :

$$(37) \quad \bar{R}_E = T \left[\frac{S+2}{2S} + \frac{\rho}{2(1-\rho)} \right]$$

Ce corollaire peut être considéré comme une conséquence du corollaire 2 par l'intermédiaire du théorème de Little. Nous allons le démontrer sans faire appel à ce résultat.

Considérons une demande de transfert C^* concernant le secteur privilégié : nous noterons τ l'instant où elle est communiquée au superviseur du disque, d l'instant du dernier passage avant τ du début du secteur privilégié sous les têtes de lecture-écriture.



(les instants où les têtes sont au-dessus du secteur privilégié constituent les intervalles hachurés)

Figure 11

Nous désignerons par :

- . L la variable aléatoire $\tau - d$
- . N_d le nombre des demandes de transfert concernant le secteur considéré qui sont encore insatisfaites à l'instant d
- . Q_d le nombre de ces demandes à l'exception (éventuellement) de celle qui est traitée entre d et $d + \frac{T}{S}$
- . K le nombre des demandes qui se manifestent dans l'intervalle de temps $]d, \tau[$

Il est clair que :

$$(38) \quad R = (T - L) + (Q + K) T + \frac{T}{S}$$

Le nombre Q est égal à

$$\begin{cases} N_d - 1 & \text{si } N_d \geq 1 \\ 0 & \text{si } N_d = 0 \end{cases}$$

Il ne dépend évidemment pas de L et

$$E(Q) = E(N_d) - \Pr [N_d \geq 1]$$

La fonction génératrice de la distribution de N_d étant la fonction du corollaire 1, cette espérance est égale à :

$$E(Q) = \left(\rho + \frac{\rho^2}{2(1-\rho)} \right) - \rho = \frac{\rho^2}{2(1-\rho)}$$

Par ailleurs, comme le processus d'arrivée est poissonnien de taux $\frac{\lambda}{S}$

$$E(K \mid t \leq L < t + dt) = \frac{\lambda}{S} t + o(dt)$$

L'égalité (38) a donc pour conséquence :

$$E(R \mid t \leq L < t + dt) = (T - t) + T \left[\frac{\rho^2}{2(1-\rho)} + \frac{\lambda}{S} t + \frac{1}{S} \right] + o(dt)$$

Admettons momentanément que la distribution de L est uniforme sur $[0, T[$, ce résultat faisant l'objet d'un lemme ci-après.

$$\begin{aligned}\bar{R}_E &= \int_0^T E(R \mid t \leq L < t + dt) \frac{dt}{T} \\ &= \frac{T}{2} + T \left[\frac{\rho^2}{2(1-\rho)} + \frac{\rho}{2} + \frac{1}{S} \right] \\ &= T \left[\frac{S+2}{2S} + \frac{\rho}{2(1-\rho)} \right]\end{aligned}$$

Lemme :

La distribution de L est uniforme sur $[0, T[$

Nommons I^* l'intervalle $[d, d+T[$ au cours duquel se produit la demande de transfert C^* . Soient t un élément de $[0, T[$ et dt un réel positif suffisamment petit pour que $t + dt < T$. Nous désignerons par E l'évènement :

$$\{t \leq L < t + dt\}$$

D'après le théorème des probabilités totales

$$\Pr[E] = \sum_{n=1}^{\infty} \Pr[E ; n \text{ arrivées de demandes au cours de } I^*]$$

Décomposons à nouveau les évènements dont les probabilités figurent dans cette somme :

$$(39) \quad \Pr[E ; n \text{ arrivées au cours de } I^*] = \sum_{i=1}^n a_i x b_i \Pr[n \text{ arrivées dans } I^*]$$

ou l'on a posé :

$$a_i = \Pr[C^* \text{ est le } i^{\text{ème}} \text{ client arrivant dans } I^* \mid n \text{ arrivées dans } I^*]$$

$$\text{et } b_i = \Pr[\text{le } i^{\text{ème}} \text{ client arrivant dans } I^* \text{ arrive entre } t \text{ et } t+dt \mid n \text{ arrivées dans } I^*]$$

La demande de transfert C^* étant quelconque, $a_i = \frac{1}{n}$. Quant à b_i c'est, au second ordre près, la probabilité que les trois évènements

$$\varepsilon_a : (i-1) \text{ arrivées dans } [0, t[$$

$$\varepsilon_b : 1 \text{ arrivée dans } [t, t+dt[$$

$$\varepsilon_c : (n-i) \text{ arrivées dans } [t+dt, T[$$

soient simultanément réalisés sachant qu'il y a n arrivées dans I^* . Les évènements ϵ_a , ϵ_b et ϵ_c étant indépendants la probabilité qu'ils soient tous réalisés est :

$$\begin{aligned}\pi &= e^{-\lambda t} \frac{(\lambda t)^{i-1}}{(i-1)!} \times \lambda dt \times e^{-\lambda(T-t-dt)} \frac{(\lambda(T-t-dt))^{n-i}}{(n-i)!} + o(dt) \\ &= e^{-\lambda T} \times \frac{1}{(i-1)!(n-i)!} (\lambda t)^{i-1} (\lambda(T-t))^{n-i} \lambda dt + o(dt)\end{aligned}$$

$$\begin{aligned}\text{donc } b_i &= \pi \times \frac{n!}{e^{-\lambda T} (\lambda T)^n} \\ &= \frac{n}{(\lambda T)^{n-1}} C_{n-1}^{i-1} (\lambda t)^{i-1} (\lambda(T-t))^{n-i} \frac{dt}{T} + o(dt)\end{aligned}$$

Reportons ce résultat dans l'égalité (39)

$$\begin{aligned}\text{Pr } [E ; n \text{ arrivées au cours de } I^*] &= \\ \frac{dt}{T} \frac{\text{Pr } [n \text{ arrivées dans } I^*]}{(\lambda T)^{n-1}} \times \sum_{i=1}^n C_{n-1}^{i-1} (\lambda t)^{i-1} (\lambda(T-t))^{n-i} \\ &= \frac{dt}{T} \text{Pr } [n \text{ arrivées dans } I^*]\end{aligned}$$

$$\text{Donc } \text{Pr } [E] = \frac{dt}{T} \sum_{n=1}^{\infty} \text{Pr } [n \text{ arrivées dans } I^*] = \frac{dt}{T}$$

Ceci achève la démonstration du lemme.

2.2.2. Résultats numériques

Nous pouvons rapprocher les résultats présents de ceux qui ont été obtenus au paragraphe 2.1.3..

Le nombre total moyen de demandes de transfert non encore acquittées est égal à $\bar{N}_{\text{PAPS}}$ si le disque est géré selon le mode premier arrivé - premier servi et à $S \times \bar{N}_E$ s'il est géré selon l'algorithme d'Eschenbach. La figure 12 représente la variation de ce nombre total en fonction de λ pour un disque à quatre secteurs. On remarque que l'algorithme d'Eschenbach est nettement plus efficace dès qu'il y a plus d'une nouvelle demande de transfert par

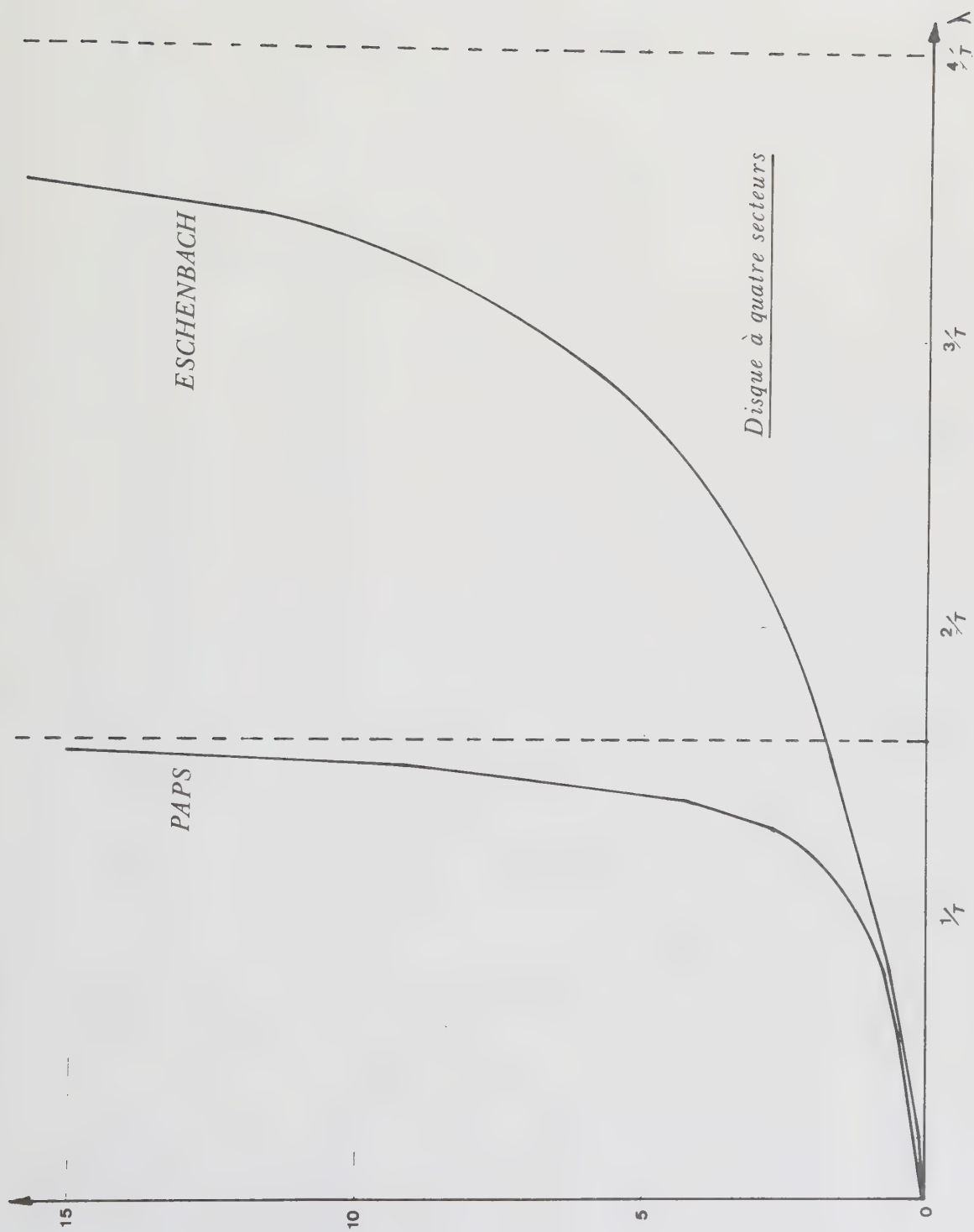


Figure 12

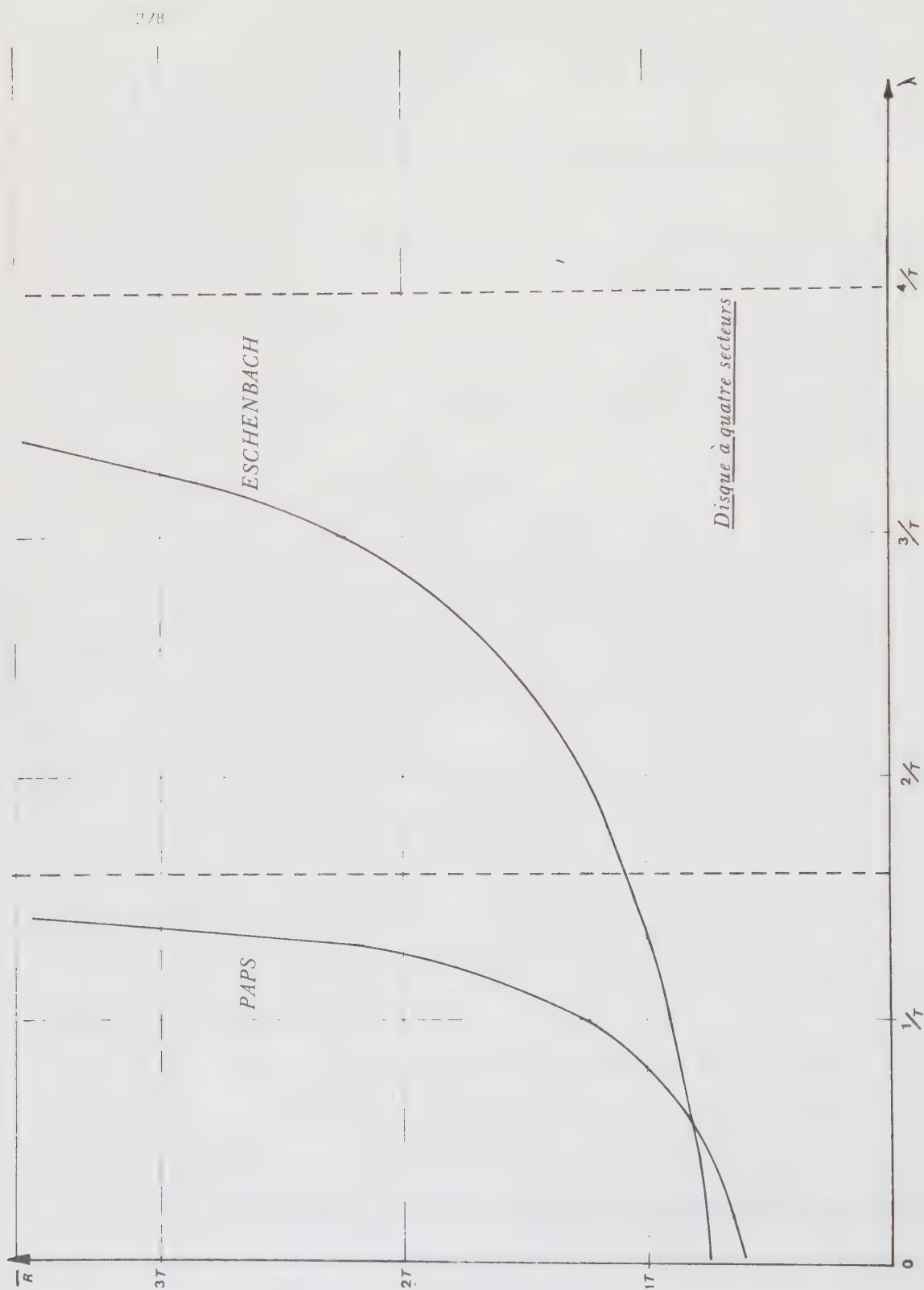


Figure 13

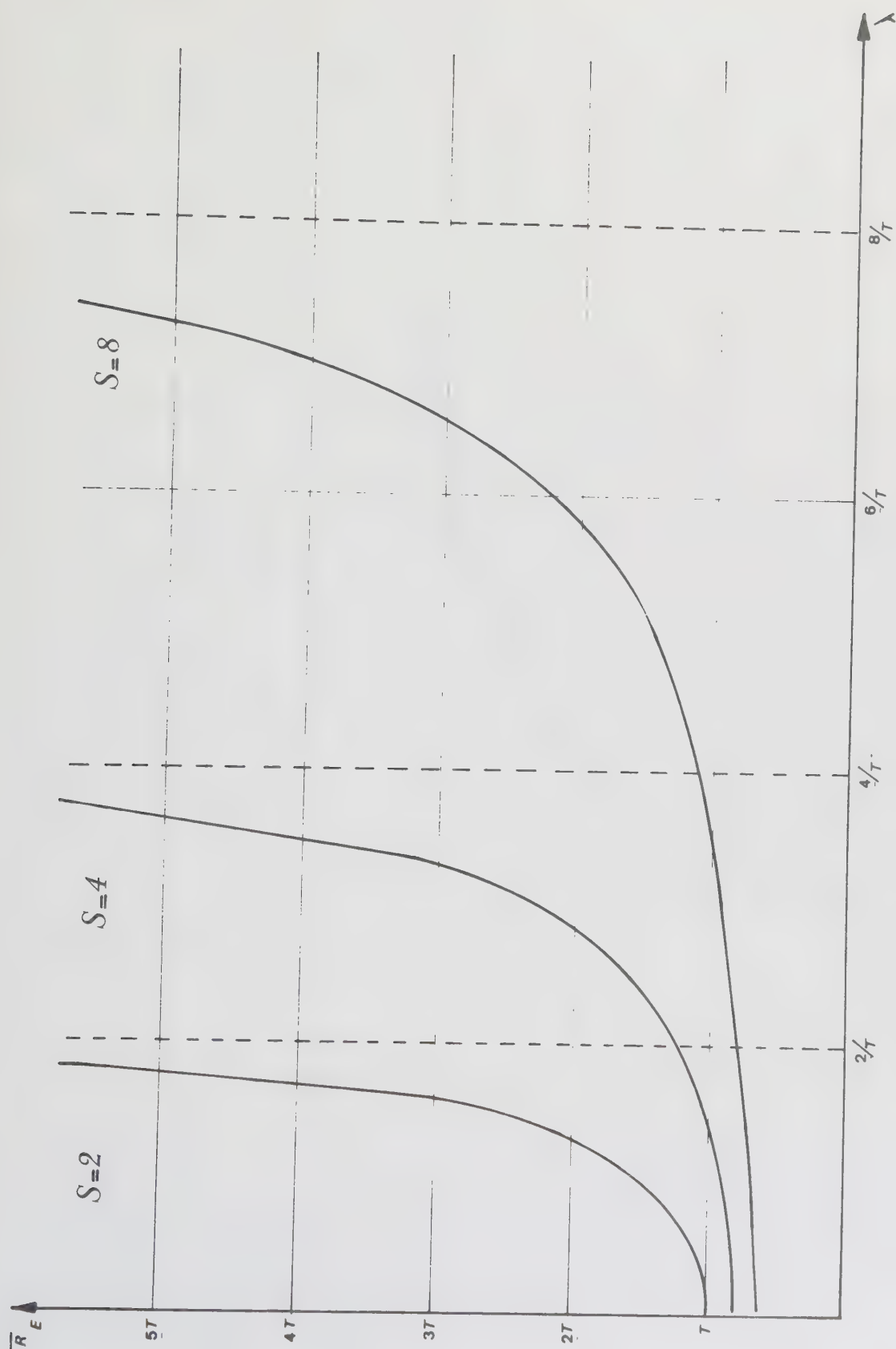


Figure 14

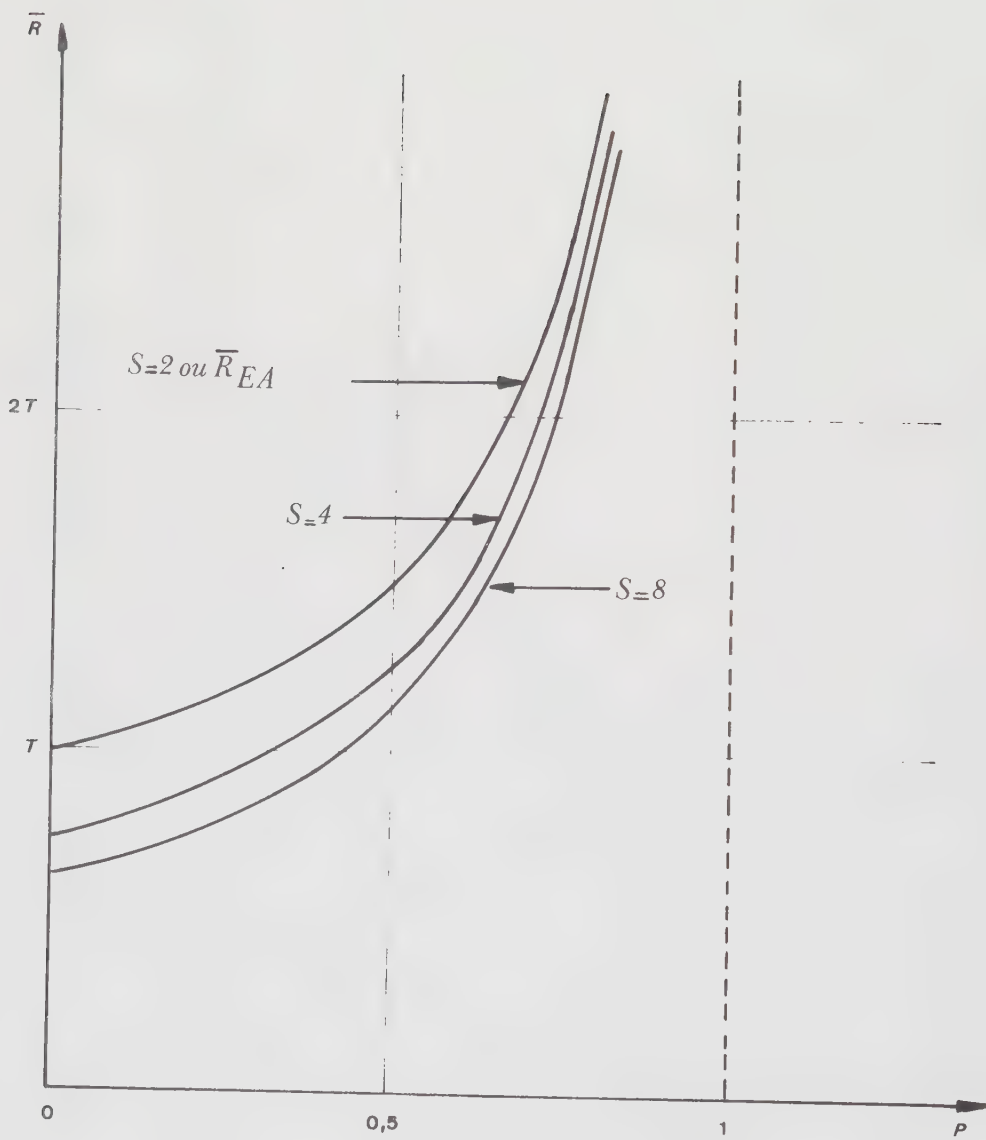


Figure 15.

tour ; en particulier il faut une charge bien plus lourde pour saturer le système que quand la politique est PAPS. Ceci se confirme sur la figure 13 qui montre la variation du temps de réponse moyen dans les mêmes conditions. Notons que si la gestion PAPS peut paraître plus efficace lorsque λ est proche de 0, cela est dû en réalité à l'approximation que nous avons faite pour le calcul de $\bar{N}_{\text{PAPS}}$ et $\bar{R}_{\text{PAPS}}$: nous avons sous-évalué le temps de service pour un transfert dont la demande arrive alors que le système est inactif.

La possibilité de traiter des flux élevés de demandes de transfert sans un accroissement prohibitif du temps de réponse est l'avantage essentiel de l'algorithme d'Eschenbach. A vitesse de rotation donnée, le nombre de secteurs joue un rôle primordial (fig. 14).

La figure 15 représente la variation du temps de réponse moyen en fonction de l'intensité de trafic $\rho = \frac{\lambda T}{S}$. Pour une valeur donnée de ρ , $\bar{R}_E$ décroît lorsque le nombre de secteurs augmente : cela suggère qu'un disque à $2S$ secteurs peut être plus efficace -du point de vue du temps de réponse moyen- que deux disques à S secteurs, installés en parallèle et tournant à la même vitesse que le premier. On remarque également que la valeur approchée $\bar{R}_{EA}$ est égale au temps de réponse $\bar{R}_E$ d'un disque à deux secteurs ; cette approximation est d'autant plus mauvaise que le nombre de secteurs est plus grand.

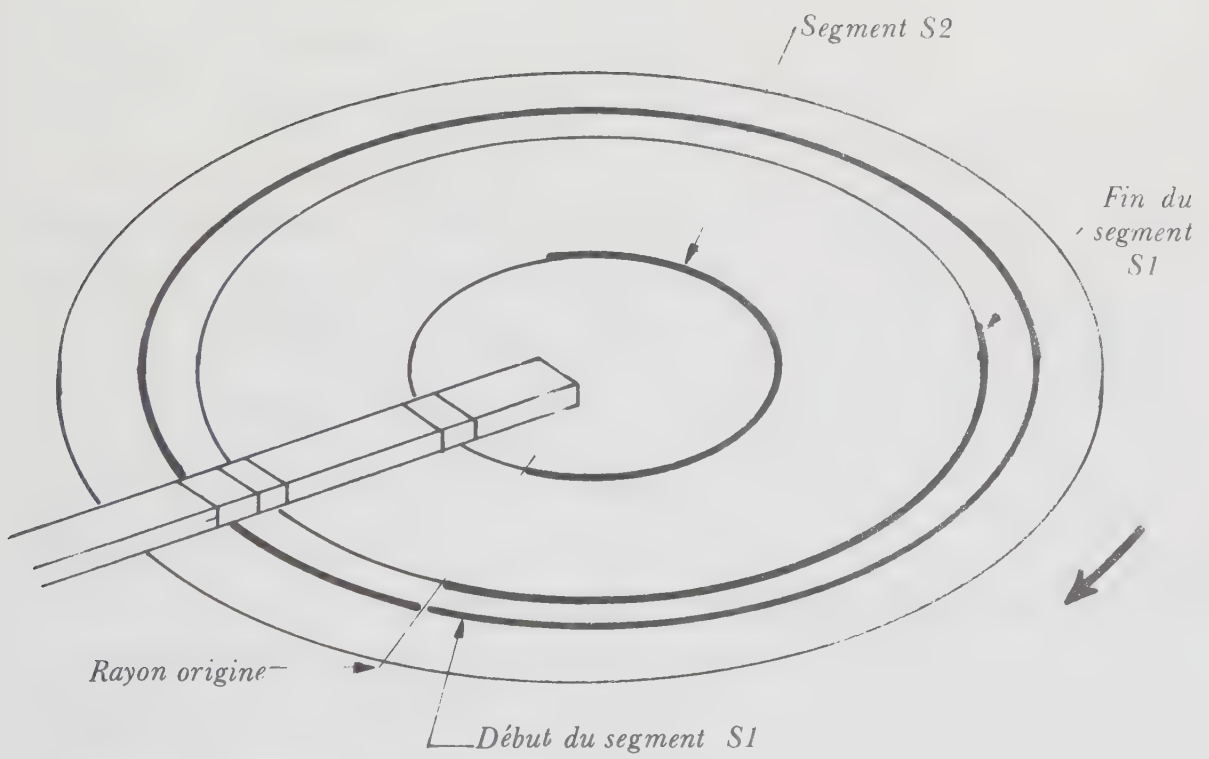
2.3 GESTION DES DISQUES A TÊTES FIXES SANS SECTEURS. [17]

Nous venons d'analyser la gestion des disques à secteurs qui sont bien adaptés à la pagination. Lorsque les transferts entre mémoire centrale et mémoire secondaire se font par segments, il peut être préférable d'utiliser des disques à têtes fixes sans secteurs. Un algorithme fréquemment utilisé dans ce cas consiste à réserver pour tous les segments des emplacements débutant sur le même rayon du disque (ou sur la même génératrice s'il s'agit d'un tambour) que nous appellerons rayon origine (voir figure 16). Les demandes de transfert sont traitées dans l'ordre des arrivées : PAPS. Lorsque plusieurs demandes sont en attente, le temps d'accès pour l'une d'elles est donc la durée entre la fin du transfert précédent et le passage sous les têtes de lecture-écriture du rayon origine. Cet algorithme, que nous désignerons sous le nom d'algorithme de l'origine commune, est facile à programmer. En outre il se révèle particulièrement efficace lorsque la taille des segments à transférer est fréquemment proche -par valeurs inférieures- de la capacité d'un nombre entier de pistes. Nous allons étudier son comportement en régime stationnaire que nous comparerons à celui de l'algorithme PAPS sans origine commune.

Dans tout ce paragraphe nous ferons les hypothèses suivantes sur les caractéristiques d'un segment à transférer. La longueur sera considérée comme une valeur d'une variable aléatoire ℓ dont la distribution est définie par la fonction de répartition :

$$F(x) = \Pr[\ell \leq x]$$

l'unité de longueur choisie étant le contenu d'une piste ; la valeur moyenne de cette variable aléatoire sera notée $\bar{\ell}$. En outre les longueurs des segments à transférer sont stochastiquement indépendantes et les arrivées de demandes ont lieu selon un processus de Poisson de taux λ . Nous serons amenés à considérer les probabilités suivantes :



Gestion d'un disque sans secteurs

Figure 16

$$. r_n = \Pr [n-1 < \ell \leq n] = F(n) - F(n-1) \quad (n = 1, 2, 3, \dots)$$

C'est la probabilité que le transfert d'un segment nécessite n rotations lorsqu'on utilise l'algorithme à origine commune. Cette suite de probabilités sera représentée par sa fonction génératrice $R(z) = \sum_{n=1}^{\infty} r_n z^n$. Nous noterons $\bar{r}$ et $E(r^2)$ les deux premiers moments de cette distribution du nombre de rotations :

$$\bar{r} = \sum_{n=1}^{\infty} n r_n \quad E(r^2) = \sum_{n=1}^{\infty} n^2 r_n$$

$$. c_u = \Pr [\ell > u \mid \ell > u-1] = \frac{1-F(u)}{1-F(u-1)} \quad (u = 1, 2, 3, \dots)$$

Lorsque u pistes d'un segment ont été transmises, la probabilité que le transfert ne soit pas encore achevé est c_u . Les suites (r_u) et (c_u) sont liées par la relation :

$$c_j = \frac{\sum_{u=j+1}^{\infty} r_u}{\sum_{u=j}^{\infty} r_u}$$

2.3.1. Analyse de l'algorithme de l'origine commune

Nous choisirons comme instant 0 un instant où le rayon origine passe sous les têtes de lecture-écriture ; nous supposerons qu'aucune requête n'est alors en cours de traitement ou en attente. Le nombre des demandes de transfert non satisfaites à l'instant t sera $N(t)$, la fonction $N(.)$ étant définie aux instants d'arrivée ou de fin de traitement de façon à être continue à droite.

Considérant les instants successifs où le rayon origine passe sous les têtes de lecture-écriture, nous sommes amenés à étudier le processus stochastique à temps discret $\mathcal{C}$ dont la valeur à l'instant qT ($q \in \mathbb{N}$) est

$$. 0 \text{ si } N(qT) = 0$$

- $(n,u) \quad (n,u \in \mathbb{N}^*)$ si $N(qT) = n > 0$ et si la révolution débutant à cet instant est la $u^{\text{ème}}$ pour le compte du transfert en cours.

Il est clair que $\mathcal{C}$ est une chaîne de Markov. La première partie de ce paragraphe sera consacrée à l'étude de sa distribution à l'équilibre. Nous utiliserons les notations suivantes :

- $p(i,t) = \Pr [N(t) = i \mid N(0) = 0] \quad (i \in \mathbb{N}, t \in \mathbb{R}^+)$
- $\bar{N}_{0c}$ désignera la valeur moyenne de $N(t)$ c'est-à-dire la limite, si elle existe, de l'expression

$$\frac{\int_0^\theta \bar{N}(t) dt}{\theta}$$

lorsque θ tend vers l'infini.

- π_0 (resp. π_{nu}) est la probabilité qu'à l'équilibre la chaîne de Markov $\mathcal{C}$ soit dans l'état 0 (resp. (n,u)) - sous réserve, évidemment que l'équilibre existe. Il est clair que

$$(40) \quad \pi_0 = \lim_{q \rightarrow \infty} p(0, qT)$$

et

$$(41) \quad \sum_{u=1}^{\infty} \pi_{nu} = \lim_{q \rightarrow \infty} p(n, qT) \quad (n \geq 1)$$

$$\bullet \quad \Pi_u(z) = \sum_{n=1}^{\infty} \pi_{nu} z^n \quad (u \geq 1, z \in [-1, 1])$$

- a_i est la probabilité de l'arrivée de i demandes d'entrée/sortie au cours d'une révolution. La fonction génératrice de cette suite est $A_T(z) = \sum_{i=0}^{\infty} a_i z^i = e^{-\lambda T(1-z)}$

Théorème

Si $\bar{r}\lambda T < 1$, la chaîne de Markov $\mathcal{C}$ est ergodique. Sa distribution à l'équilibre est telle que

$$(42) \quad \pi_0 = 1 - \bar{r}\lambda T$$

$$(43) \quad \pi_1(z) = \pi_0 z \frac{e^{-\lambda T(1-z)} - 1}{z - R(e^{-\lambda T(1-z)})} \quad (z \neq 1)$$

$$(44) \quad \pi_u(z) = (1 - F(u-1)) e^{-\lambda T(u-1)(1-z)} \pi_1(z)$$

Calculons les probabilités de transition de la chaîne de Markov $\mathcal{X}$.

Nous pouvons distinguer plusieurs types de transitions :

$$\begin{aligned} \cdot \Pr [0 \xrightarrow{\quad} 0] &= \Pr [0 \text{ arrivée dans le système au cours d'une révolution}] \\ &= a_0 \end{aligned}$$

· Pour tout entier $n \geq 1$

$$\Pr [0 \xrightarrow{\quad} (n,1)] = \Pr [n \text{ arrivées au cours d'une révolution}] = a_n$$

· Pour tous les triplets d'entiers n, u, j tels que

$$n \geq 1, u \geq 2, j \geq 0, j \leq n-1 :$$

$$\begin{aligned} \Pr [(n-j, u-1) \xrightarrow{\quad} (n,u)] &= \Pr [j \text{ arrivées et non achèvement du transfert} \\ &\quad \text{en cours durant la révolution}] \\ &= \Pr [j \text{ arrivées durant une révolution}] \times \Pr [\ell > u-1 \mid \ell > u-2] \\ &= a_j \cdot c_{u-1} \end{aligned}$$

· Pour tous les triplets d'entiers n, u, j tels que

$$n \geq 1, u \geq 1, j \geq 0, j \leq n :$$

$$\begin{aligned} \Pr [(n-j+1, u) \xrightarrow{\quad} (n,1)] &= \Pr [j \text{ arrivées pendant une révolution}] \\ &\quad \times \Pr [\ell \leq u \mid \ell > u-1] \\ &= a_j (1 - c_u) \end{aligned}$$

· Pour tout entier $u \geq 1$

$$\begin{aligned} \Pr [(1, u) \xrightarrow{\quad} 0] &= \Pr [0 \text{ arrivée pendant une révolution}] \times \\ &\quad \Pr [\ell \leq u \mid \ell > u-1] \\ &= a_0 (1 - c_u) \end{aligned}$$

· Toutes les autres transitions ont une probabilité nulle de se réaliser.

Le graphe des transitions est donc fortement connexe et apériodique (il possède des boucles). La distribution à l'équilibre, si elle existe, satisfera aux équations :

$$(E^*) \quad \pi_0 = a_0 \pi_0 + \sum_{u=1}^{\infty} a_0 (1-c_u) \pi_{1u}$$

$$(E_1^n) \quad \pi_{n1} = a_n \pi_0 + \sum_{j=0}^n \sum_{u=1}^{\infty} a_j (1-c_u) \pi_{n-j+1 u} \quad (n \in \mathbb{N})$$

$$(E_u^n) \quad \pi_{nu} = \sum_{j=0}^{n-1} a_j c_{u-1} \pi_{n-j u} \quad (n \geq 0, u \geq 2)$$

Multiplions les deux membres de l'équation E^* (resp. E_1^n) par z (resp. z^{n+1}) et sommons membre à membre les égalités ainsi obtenues :

$$(45) \quad z(\pi_0 + \Pi_1(z)) = \pi_0 z A_T(z) + \sum_{u=1}^{\infty} (1-c_u) A_T(z) \Pi_u(z)$$

De même, en multipliant par z^n les deux membres de l'équation E_u^n et en sommant pour une même valeur de u , nous obtenons :

$$\Pi_u(z) = c_{u-1} A_T(z) \Pi_{u-1}(z) \quad (u = 2, 3, 4, \dots)$$

Ce qui, compte tenu de l'égalité $c_u = \frac{1-F(u)}{1-F(u-1)}$ est équivalent à :

$$\Pi_u(z) = \frac{1-F(u-1)}{1-F(u-2)} \times \frac{1-F(u-2)}{1-F(u-3)} \times \dots \times \frac{1-F(1)}{1-F(0)} A_T^{u-1}(z) \Pi_1(z)$$

c'est-à-dire à

$$(46) \quad \Pi_u(z) = (1-F(u-1)) A_T^{u-1}(z) \Pi_1(z) \quad (u = 2, 3, 4, \dots)$$

Utilisons ce résultat dans l'égalité (45) en remarquant que $1-c_u = \frac{F(u)-F(u-1)}{1-F(u-1)}$

$$\begin{aligned} z(\pi_0 + \Pi_1(z)) &= \pi_0 z A_T(z) + \sum_{u=1}^{\infty} (F(u) - F(u-1)) A_T^u(z) \Pi_1(z) \\ &= \pi_0 z A_T(z) + \sum_{u=1}^{\infty} r_u A_T^u(z) \Pi_1(z) \\ &= \pi_0 z A_T(z) + R(A_T(z)) \Pi_1(z) \end{aligned}$$

Donc

$$(47) \quad \Pi_1(z) = \pi_0 z \frac{1-A_T(z)}{R(A_T(z))-z} \quad (z \neq 1)$$

Pour déterminer π_0 , nous allons utiliser le fait que la somme des probabilités est égale à 1, c'est-à-dire que

$$\pi_0 + \sum_{u=1}^{\infty} \Pi_u(1) = 1$$

ce qui, selon (46), implique :

$$\begin{aligned} 1 &= \pi_0 + \Pi_1(1) \times \sum_{u=1}^{\infty} (1-F(u-1)) \\ &= \pi_0 + \Pi_1(1) \sum_{u=1}^{\infty} \sum_{n=u}^{\infty} r_n \\ (48) \quad &= \pi_0 + r \Pi_1(1) \end{aligned}$$

L'égalité (47) nous fournit une autre relation entre π_0 et $\Pi_1(1)$:

$$\Pi_1(1) = \pi_0 \lim_{z \rightarrow 1} \frac{A_T(z) - 1}{z - R(A_T(z))}$$

soit, par application de la règle de l'Hospital :

$$(49) \quad \Pi_1(1) = \pi_0 \frac{\lambda T}{1 - \bar{r} \lambda T}$$

De la confrontation de (48) et (49), il résulte

$$\pi_0 = 1 - \bar{r} \lambda T$$

La condition d'ergodicité de la chaîne de Markov $\mathcal{X}$ est donc $\bar{r} \lambda T < 1$. Les autres affirmations du théorème sont contenues dans les égalités (46) et (47).

Corollaire 1

Si $\bar{r} \lambda T < 1$, la distribution à l'équilibre du nombre de demandes de transfert insatisfaites à l'instant où le rayon origine passe sous les têtes de lecture a pour fonction génératrice :

$$(50) \quad \Pi(z) = \pi_0 + \sum_{u=1}^{\infty} \Pi_u(z) = (1 - r \lambda T) (1 - z) \frac{R(e^{-\lambda T(1-z)}) - 1}{R(z^{-\lambda T(1-z)}) - z} \quad (z \neq 1)$$

De l'égalité (44) il résulte en effet :

$$\begin{aligned}\Pi(z) &= \pi_0 + \Pi_1(z) \sum_{u=1}^{\infty} (1-F(u-1)) (e^{-\lambda T(1-z)})^{u-1} \\ &= \pi_0 + \Pi_1(z) \sum_{u=1}^{\infty} \sum_{j=u}^{\infty} r_j (e^{-\lambda T(1-z)})^{u-1}\end{aligned}$$

En modifiant l'ordre de la sommation, nous obtenons :

$$\Pi(z) = \pi_0 + \Pi_1(z) \sum_{k=1}^{\infty} r_k \sum_{u=1}^k (e^{-\lambda T(1-z)})^{u-1}$$

c'est-à-dire, si $z \neq 1$:

$$\begin{aligned}\Pi(z) &= \pi_0 + \Pi_1(z) \sum_{k=1}^{\infty} r_k \frac{(e^{-\lambda T(1-z)})^k - 1}{e^{-\lambda T(1-z)} - 1} \\ &= \pi_0 + \Pi_1(z) \frac{R(e^{-\lambda T(1-z)}) - 1}{e^{-\lambda T(1-z)} - 1}\end{aligned}$$

Il suffit de reporter dans cette égalité les valeurs précédemment trouvées de π_0 et $\Pi_1(z)$ pour achever la démonstration du corollaire.

Corollaire 2

Si $\bar{r}\lambda T < 1$, la valeur moyenne du nombre de demandes de transfert non encore satisfaites, c'est-à-dire la limite lorsque θ tend vers l'infini de $\frac{1}{\theta} \int_0^\theta N(t)dt$ est égale à

$$(51) \quad \bar{N}_{OC} = \lambda T \left(l + \frac{1}{2} \right) + \frac{(\lambda T)^2}{2} \frac{E(r^2)}{1 - \bar{r}\lambda T}$$

Supposons que la distribution de la chaîne de Markov $\mathcal{C}$ à l'instant qT ($q \in \mathbb{N}$) soit la distribution d'équilibre. Désignons par $p_n(x)$ la probabilité qu'il y ait n demandes insatisfaites à l'instant $qT + xT$ ($0 \leq x < 1$) ; la fonction génératrice de la distribution ainsi définie est :

$$P(z, x) = \sum_{n=0}^{\infty} p_n(x) z^n$$

Par un raisonnement analogue à celui qui est employé pour la démonstration du corollaire 2 au paragraphe 2.2.1., il n'est pas difficile de montrer que :

$$(52) \quad N_{0C} = \int_0^1 \frac{\partial P}{\partial z} (1, x) dx$$

Notons $a_j(xT)$ la probabilité de j arrivées suivant un processus de Poisson de taux λ , au cours d'un intervalle de temps de durée xT . La probabilité qu'une longueur de segment supérieure à $u-1$ soit inférieure à $u-1+x$ (c'est-à-dire $\Pr[\ell \leq u-1+x \mid \ell > u-1]$) est égale à $\frac{F(u-1+x) - F(u-1)}{1 - F(u-1)}$. Il en résulte, pour tout entier n positif ou nul :

$$\begin{aligned} p_n(x) = & \sum_{u=1}^{\infty} \left(1 - \frac{F(u-1+x) - F(u-1)}{1 - F(u-1)}\right) \sum_{j=1}^n a_{n-j}(xT) \pi_{ju} \\ & + \pi_0 a_n(xT) + \sum_{u=1}^{\infty} \frac{F(u-1+x) - F(u-1)}{1 - F(u-1)} \sum_{j=0}^n a_{n-j}(xT) \pi_{j+1} \end{aligned}$$

($n = 0, 1, 2, \dots$)

Multiplions les deux membres de cette égalité par z^n et sommons pour toutes les valeurs de n :

$$\begin{aligned} P(z, x) = & \sum_{u=1}^{\infty} \left(1 - \frac{F(u-1+x) - F(u-1)}{1 - F(u-1)}\right) A_{xT}(z) \Pi_u(z) \\ & + \pi_0 A_{xT}(z) + \sum_{u=1}^{\infty} \frac{F(u-1+x) - F(u-1)}{1 - F(u-1)} x \frac{A_{xT}(z) \Pi_u(z)}{z} \end{aligned}$$

Utilisons ce résultat pour évaluer l'expression (52). Après quelques calculs intermédiaires nous obtenons :

$$\bar{N}_{0C} = \frac{\lambda T}{2} - \int_0^1 \sum_{u=1}^{\infty} \frac{F(u-1+x) - F(u-1)}{1 - F(u-1)} \Pi_u(1) dx + \sum_{u=1}^{\infty} \Pi'_u(1)$$

soit, moyennant le théorème que nous venons de démontrer :

$$(53) \quad \bar{N}_{0C} = \frac{\lambda T}{2} - \Pi_1(1) \int_0^1 \sum_{u=1}^{\infty} (F(u-1+x) - F(u-1)) dx + \sum_{u=1}^{\infty} \Pi'_u(1)$$

L'intégrale $\int_0^1 \sum_{u=1}^{\infty} (F(u-1+x) - F(u-1)) dx$ est égale à

$$\begin{aligned} \sum_{u=1}^{\infty} \int_{u-1}^u (F(x) - F(u-1)) dx &= \sum_{u=1}^{\infty} \left[\int_{u-1}^u (F(x)-1) dx + 1-F(u-1) \right] \\ &= - \int_0^{\infty} (1-F(x)) dx + \sum_{u=1}^{\infty} (1-F(u-1)) = -\bar{\ell} + \bar{r} \end{aligned}$$

Par ailleurs $\sum_{u=1}^{\infty} \Pi'_u(1)$ est la valeur pour $z = 1$ de la dérivée de la fonction Π qui fait l'objet du corollaire 1. A l'aide de développements limités nous obtenons :

$$\Pi'(1) = r\lambda T + \frac{(\lambda T)^2}{2} \frac{E(r^2)}{1-r\lambda T}$$

Grâce à ces deux résultats et compte tenu de ce que $\Pi_1(1) = \lambda T$ l'égalité (53) se dérive en :

$$\bar{N}_{OC} = \lambda T \left(\bar{\ell} + \frac{1}{2} \right) + \frac{(\lambda T)^2}{2} \frac{E(r^2)}{1-r\lambda T}$$

ce qui achève la démonstration.

Corollaire 3

A l'équilibre le temps de réponse moyen est

$$\bar{R}_{OC} = T \left[\bar{\ell} + \frac{1}{2} + \frac{\lambda T}{2} \frac{E(r^2)}{1-r\lambda T} \right]$$

Nous pouvons déduire ce corollaire du précédent par application du théorème de Little ou par un calcul immédiat analogue à celui qui a été présenté dans l'analyse de la politique d'Eschenbach.

2.3.2. Résultats numériques

La gestion du disque analysée dans le paragraphe précédent est à comparer à la gestion PAPS avec emplacement quelconque de l'origine des segments. Supposons que la position angulaire par rapport au disque des origines de segments est uniformément distribuée et qu'il n'y a pas de corrélation entre les positions des segments dont le transfert est demandé. Un modèle pour cette gestion du disque est la file M/G/1 dont le taux d'arrivée est λ et dont le temps de service V est la somme de deux variables aléatoires indépendantes :

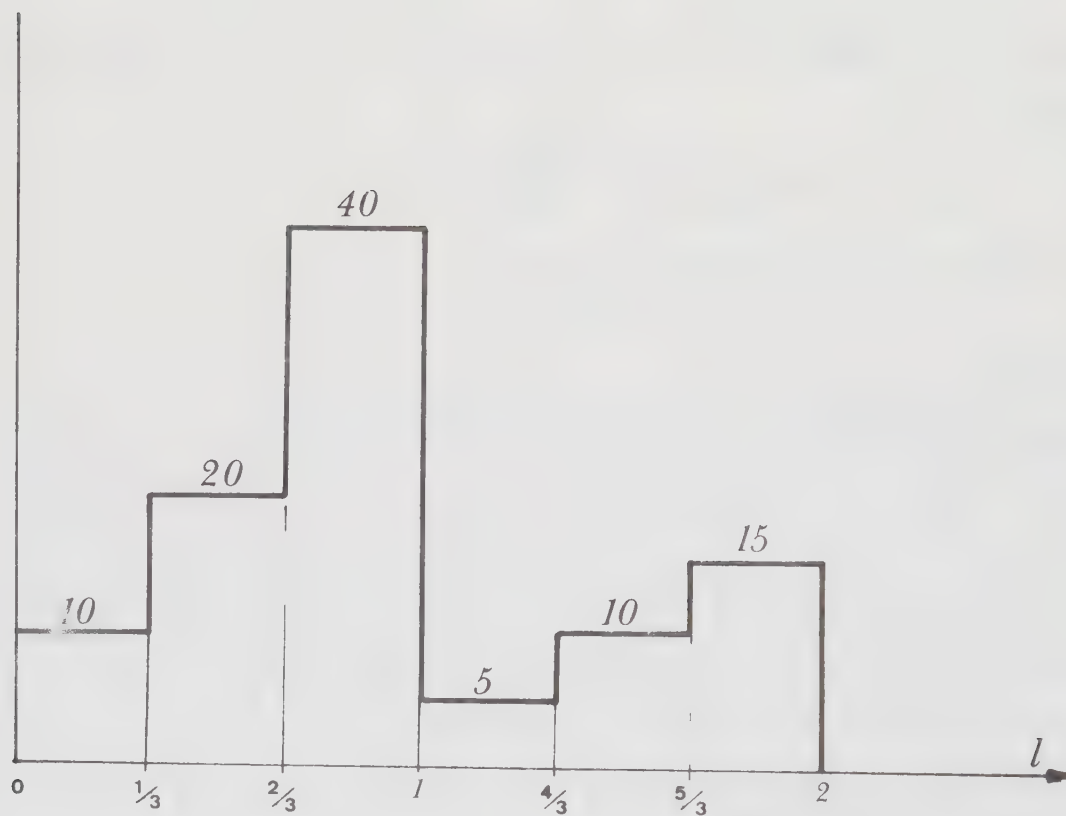


Figure 17

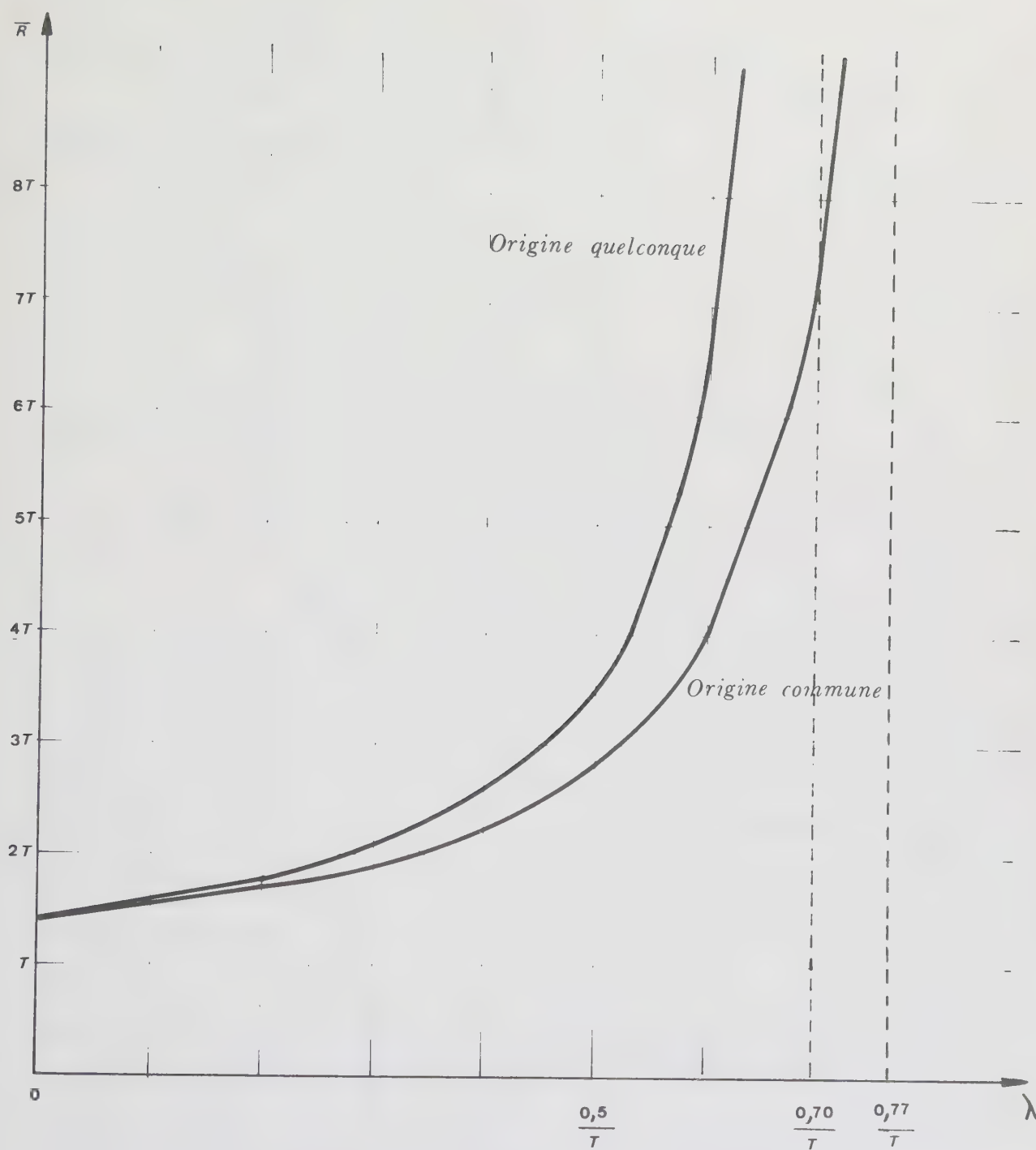


Figure 18

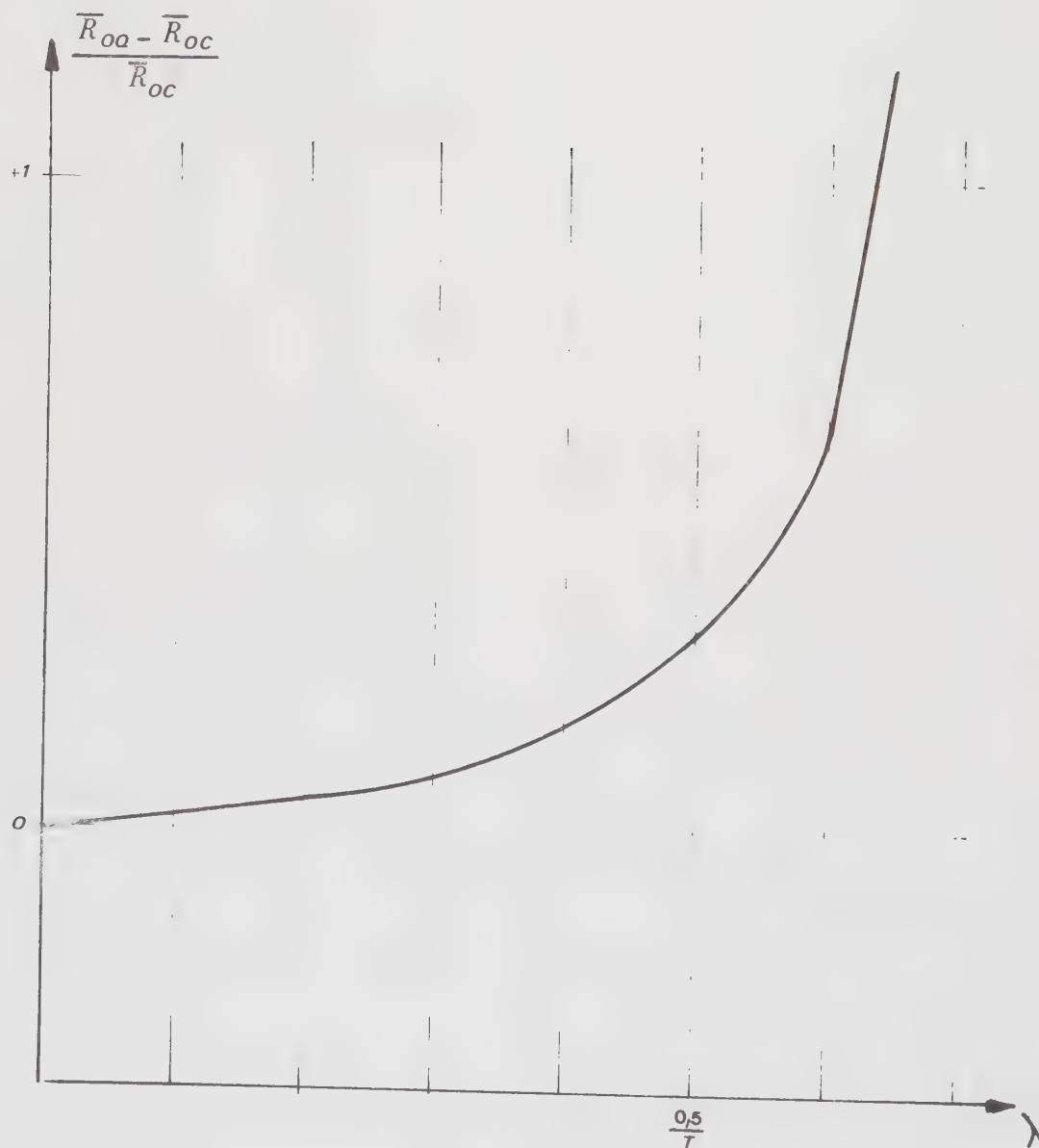


Figure 19

- A , temps d'accès, uniformément distribué sur $]0T]$
- λT , temps nécessaire au transfert proprement dit.

Appliquons à ce système la formule de Pollaczek et Khintchine.

Les deux premiers moments du temps de service sont :

$$E(V) = T\left(\frac{1}{2} + \bar{\ell}\right)$$

et

$$\begin{aligned} E(V^2) &= E(A^2) + T^2 E(\ell^2) + 2TE(A) E(\ell) \\ &= T^2\left(\frac{1}{3} + E(\ell^2) + \bar{\ell}\right) \end{aligned}$$

Le nombre moyen de segments en attente ou en cours de transfert et le temps de réponse moyen du disque sont donc respectivement :

$$\bar{N}_{OQ} = \lambda T \left(\frac{1}{2} + \bar{\ell}\right) + (\lambda T)^2 \frac{(\bar{\ell} + \frac{1}{3} + E(\ell^2))}{2(1 - \lambda T(\frac{1}{2} + \bar{\ell}))}$$

et

$$\bar{R}_{OQ} = T \left(\frac{1}{2} + \bar{\ell} + \lambda T \frac{(\bar{\ell} + \frac{1}{3} + E(\ell^2))}{2(1 - \lambda T(\frac{1}{2} + \bar{\ell}))}\right)$$

Il est intuitif que l'algorithme de l'origine commune sera le plus efficace des deux si la longueur des segments est fréquemment proche par valeurs inférieures d'un nombre entier de pistes. Pour illustrer cette comparaison nous allons supposer que la densité de répartition de ℓ est la fonction constante par morceaux représentée sur la figure 17 (les nombres portés sur cette figure sont proportionnels aux valeurs de la densité : chacun représente le pourcentage des segments à transférer dont la longueur est incluse dans le domaine de valeurs qui lui est associé). La variation de $\bar{R}_{OC}$ et $\bar{R}_{OQ}$ en fonction du taux d'arrivée est représenté sur la figure 18. Le gain qu'apporte l'utilisation de l'algorithme de l'origine commune n'est pas négligeable (fig. 19).

3 - L'UTILISATION DES DISQUES A TETES MOBILES

Dans ce paragraphe nous allons étudier la gestion du sous-système formé d'un coupleur auquel est reliée une unité de disques à têtes mobiles. La pile de disques a C cylindres, numérotés de 0 à C-1, et chacune de ses pistes est divisée en S secteurs. Comme nous l'avons noté dans l'introduction, le temps total T_S de traitement d'une requête est la somme de trois termes : T_1 , temps d'accès au cylindre, T_2 , délai de rotation et T_3 , temps de transfert qui, pour un matériel déterminé ne dépend guère que de la longueur de l'enregistrement à lire ou à écrire. Nous supposerons que tous les enregistrements sont de la taille d'un secteur de piste. Plusieurs méthodes ont été proposées pour optimiser la valeur moyenne de T_S ; elles visent à réduire T_1 ou T_2 , plus rarement les deux à la fois.

3.1 ORDONNANCEMENT DES REQUETES.

La plupart des politiques d'ordonnancement sont des variantes de celles énumérées ci-dessous.

. FCFS (First Come, First Served) est la politique la plus simple : les requêtes sont traitées selon leur ordre d'apparition. Il n'est pas tiré avantage des relations qui peuvent exister entre les diverses demandes en attente ni de la position du bras de lecture-écriture par rapport au disque.

. SSTF (Shortest Seek Time First) : lorsqu'une entrée/sortie s'achève, cet algorithme choisit comme prochaine requête à traiter celle qui est relative au cylindre le plus proche de celui au-dessus duquel se trouvent alors les têtes. Le débit est accru par rapport à ce qu'il est avec FCFS mais un fort écart-type pour la distribution du temps de réponse total est à craindre. L'étude qui suit nous montrera que cette discrimination envers certaines demandes individuelles n'est pas très importante lorsque la charge est lourde et uniformément répartie sur la surface du

disque. Un tel résultat n'est pas pour choquer l'intuition. En effet dans de telles conditions, si le bras se déplace dans une direction déterminée, la densité des requêtes sera plus élevée devant lui que derrière. Exceptionnellement l'arrivée d'une nouvelle demande pourra inverser le sens du déplacement mais il y a de fortes chances pour que le mouvement du bras reprenne ensuite sa direction initiale jusqu'à ce que le bord du disque soit atteint. Aucune zone - et en particulier pas le centre - ne semble donc devoir être favorisée.

. SCAN : le bras se déplace d'un cylindre extrême à l'autre, s'arrêtant pour servir au passage toutes les requêtes concernant des cylindres survolés. Lorsque cette méthode d'accès est employée, aucune demande n'est soumise à une discrimination abusive.

. C-SCAN (Circular SCAN) diffère de la politique précédente par le fait que les requêtes ne sont traitées que lors du balayage du cylindre 0 vers le cylindre C-1. Le retour du cylindre C-1 vers le cylindre 0 est ininterrompu.

. LOOK et C-LOOK sont des variantes respectivement de SCAN et C-SCAN. La différence est que le mouvement du bras s'inverse lorsque la dernière requête dans le sens courant (et non plus le cylindre extrême) a été atteinte. On peut penser que les deux types d'algorithmes ont des performances voisines à forte charge.

. N-step SCAN diffère de SCAN par la recherche d'une faible variance du temps de réponse. Les demandes sont réunies par groupes d'au plus N éléments. Lorsque le service d'un tel groupe s'achève, les plus anciennement arrivées des demandes, jusqu'à concurrence de N, constituent le groupe suivant. A l'intérieur de ce groupe, les requêtes sont ordonnées de sorte que la durée moyenne du déplacement du bras soit la plus faible possible : le problème à résoudre est celui du "voyageur de commerce"

(recherche d'un circuit hamiltonien de longueur minimale dans un graphe). Plus simplement on peut traiter les demandes d'un groupe en commençant par un balayage dans celle des deux directions où la distance jusqu'à la demande la plus éloignée est la plus faible puis achever le traitement par un balayage en sens inverse : ce procédé est équivalent au précédent si le temps d'accès à un cylindre est une fonction linéaire du nombre de cylindres traversés.

. ESCHENBACH. Sous cette politique de gestion, les têtes de lecture/écriture survolent pendant une révolution complète chaque cylindre depuis celui qui est numéroté 0 jusqu'à celui qui est numéroté C-1. L'arrêt du bras a lieu même au-dessus de cylindres non touchés par des demandes de transfert. Les demandes qui concernent un même cylindre sont triées par secteur ce qui permet une optimisation du délai de rotation. Toutefois à chaque balayage une requête au plus sera traitée par secteur de cylindre. Comme pour C-SCAN, le retour du cylindre C-1 au cylindre 0 est ininterrompu.

Ainsi que nous l'avons signalé dans l'introduction l'analyse de ces méthodes de gestion est délicate. Les distributions de probabilités caractérisant le comportement stationnaire sont difficiles à calculer rigoureusement et seuls des résultats partiels ont été obtenus [12] [13] [14] [15]. Nous allons donc présenter une étude comparative par simulation.

3.2 COMPARAISON DES POLITIQUES D'ORDONNANCEMENT SELON T.J. TEOREY ET T.B. PINKERTON.

Dans [5] TEOREY et PINKERTON ont publié les résultats d'une étude par simulation du comportement d'une unité de disques à têtes mobiles. Pour la variation du temps d'accès en fonction du nombre de cylindres traversés, une approximation linéaire de la courbe de la figure 7 a été choisie. Les autres caractéristiques du modèle sont aussi celles des disques IBM 2314 :

200 cylindres et période de révolution de 25 ms. Les hypothèses suivantes sont faites sur l'organisation du système et les arrivées de requêtes :

- . Un seul module de disques est considéré
- . Il y a une tête mobile par surface utile, tous les bras étant solidaires
- . L'unité de liaison et le canal sont toujours disponibles pour le module de disques
- . Les enregistrements à transférer, de la taille d'un secteur de piste, sont distribués uniformément sur toute la pile de disques
- . Les requêtes arrivent selon un processus de Poisson de taux λ
- . Aucune distinction n'est faite entre les lectures et les écritures
- . Le temps de mise en oeuvre des algorithmes d'ordonnancement est négligé.

Les figures 20 à 25 représentent la variation du temps moyen de service T_S , du temps moyen de réponse R et du carré du coefficient de variation σ_R^2 / R^2 . Pour ces courbes le nombre de secteurs par piste est fixé à $S=50$, cette valeur élevée permettant de donner une importance prépondérante au temps d'accès au cylindre et au délai de rotation.

Ces résultats montrent que la politique FCFS est satisfaisante lorsque la fréquence des demandes de transfert est faible. Elle conduit rapidement à la saturation ce qui explique qu'elle soit rarement utilisée.

SCAN et SSTF offrent d'excellents débits et temps de réponse au prix toutefois d'une variance assez élevée. Il est à remarquer que la variance est plus élevée pour SSTF que pour SCAN cette dernière politique semblant préférable sauf pour les taux de requêtes les plus faibles. L'algorithme N step SCAN est intéressant car il offre des performances moyennes raisonnables et une variance moins élevée que celle de SSTF ou de SCAN.

Grâce à l'optimisation du délai de rotation, la politique d'ESCHENBACH a de bonnes performances lorsque la charge est élevée. Remarquons que si aucune file d'attente ne se constitue au niveau des secteurs de cylindres, le temps de réponse à une requête est au plus égal à la durée d'un aller et retour du bras, ce cycle comportant C arrêts. Dans une gestion SCAN, le temps de réponse est également inférieur à un cycle mais celui-ci peut comporter jusqu'à $(2C-2)$ arrêts. Aussi la variance du temps de réponse est elle plus forte pour SCAN que pour ESCHENBACH lorsque les entrées-sorties sont demandées à un rythme élevé. Nous pouvons nous attendre à ce que sous ces conditions de charge, la méthode C-SCAN se révèle supérieure à SCAN.

Le classement des diverses politiques n'étant pas le même selon que l'on choisit pour critère T_S , R ou σ_R , il serait intéressant de disposer d'un critère global. Dans l'étude citée sont utilisées des fonctions de la forme :

$$\theta = T_S^a (b R + c \sigma_R)$$

où a , b et c sont des coefficients à choisir. Le problème est donc de trouver la politique qui à l'intérieur d'un domaine $\lambda_1 \leq \lambda \leq \lambda_2$ minimise θ . Les figures 26 à 28 représentent les valeurs de θ déduites d'une simulation. Sur ces figures apparaissent des variantes "optimisées" qui diffèrent des politiques de base par l'adjonction d'une optimisation du délai de rotation : comme dans la politique d'ESCHENBACH les requêtes sont triées par secteur. Le gain ainsi obtenu est négligeable pour N-step SCAN car l'optimisation n'est évidemment faite qu'à l'intérieur d'un groupe ; en revanche il est notable pour SCAN et C-SCAN.

Il ressort de cette étude que dans les conditions précisées SCAN (ou LOOK) est la meilleure des politiques couramment utilisées, lorsque l'utilisation du disque est faible ; en revanche C-SCAN (C-LOOK) est préférable lorsque le rythme des demandes est élevé. Si $S = 50$, la démarcation

entre les deux domaines se place à $\lambda = 30$. Comme les performances sont sensiblement équivalentes dans le domaine 28-32, il est possible de définir deux points de raccordement entre LOOK et C-LOOK. Sur le plan de la réalisation, il est plus facile d'observer la longueur de la file d'attente que le taux λ . Par simulation on détermine la longueur moyenne de la file d'attente pour les taux de 28 et 32 requêtes par seconde. La méthode de gestion que suggère cette étude est définie par le tableau ci-dessous :

Région	Taux des demandes de lecture-écriture	Politique	Longueur de la file d'attente entraînant un changement de politique
1	0-32 requêtes par seconde	LOOK	$L_2 = 140$ (longueur moyenne de la file pour LOOK à $\lambda = 32$)
2	28-50 requêtes par seconde	C-LOOK	$L_1 = 100$ (longueur moyenne de la file pour C-LOOK à $\lambda = 28$)

La figure 29 illustre la méthode des deux points de jonction.

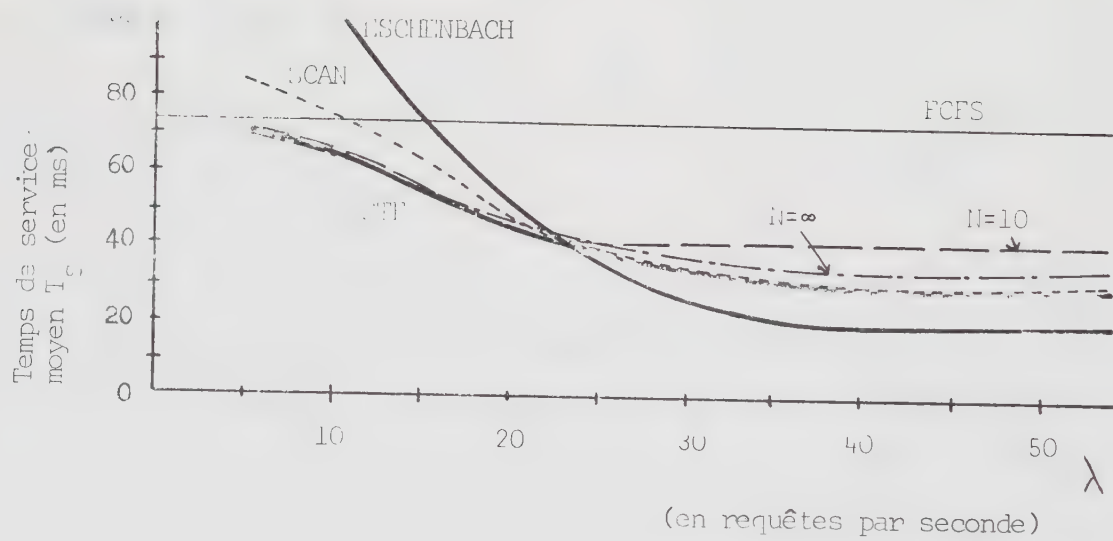


Figure 20

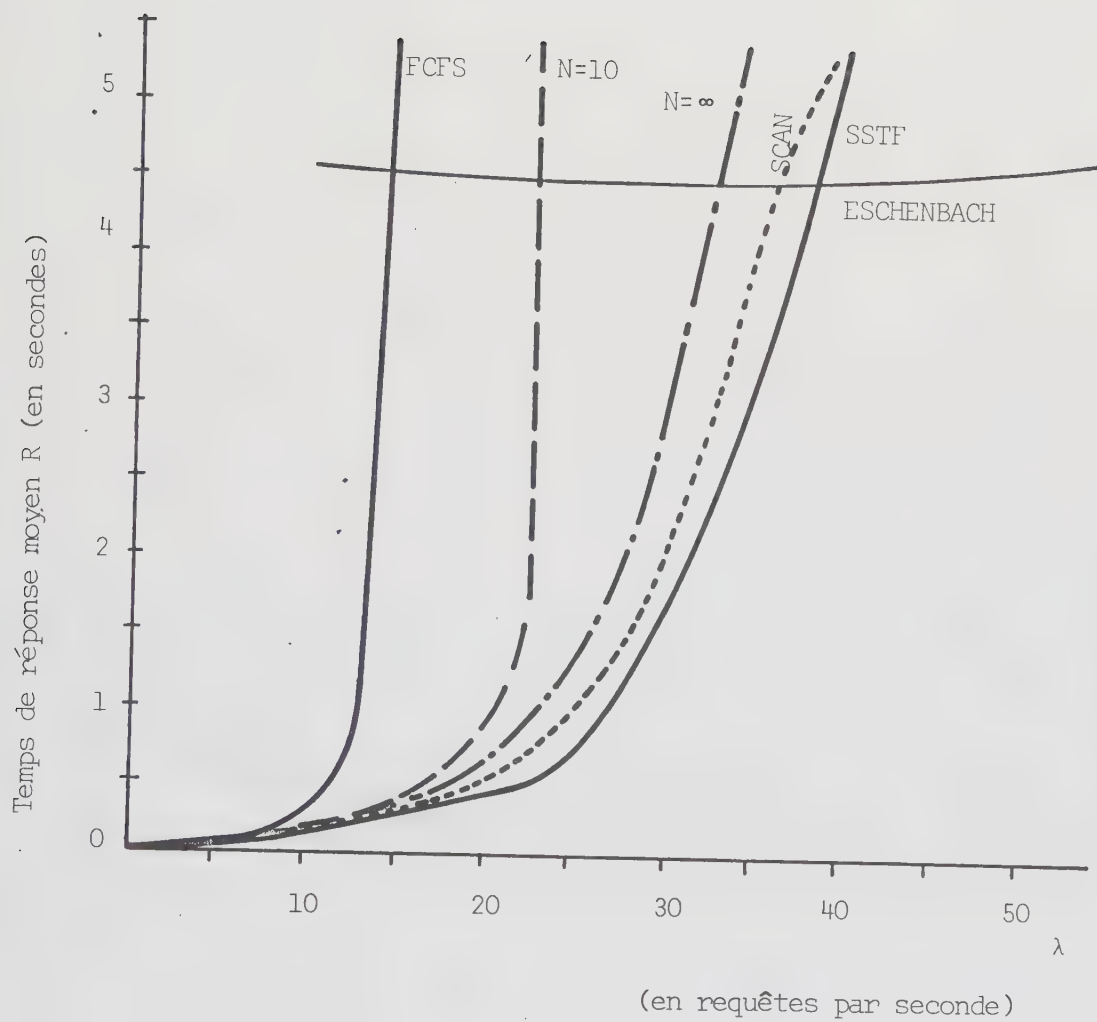


Figure 21

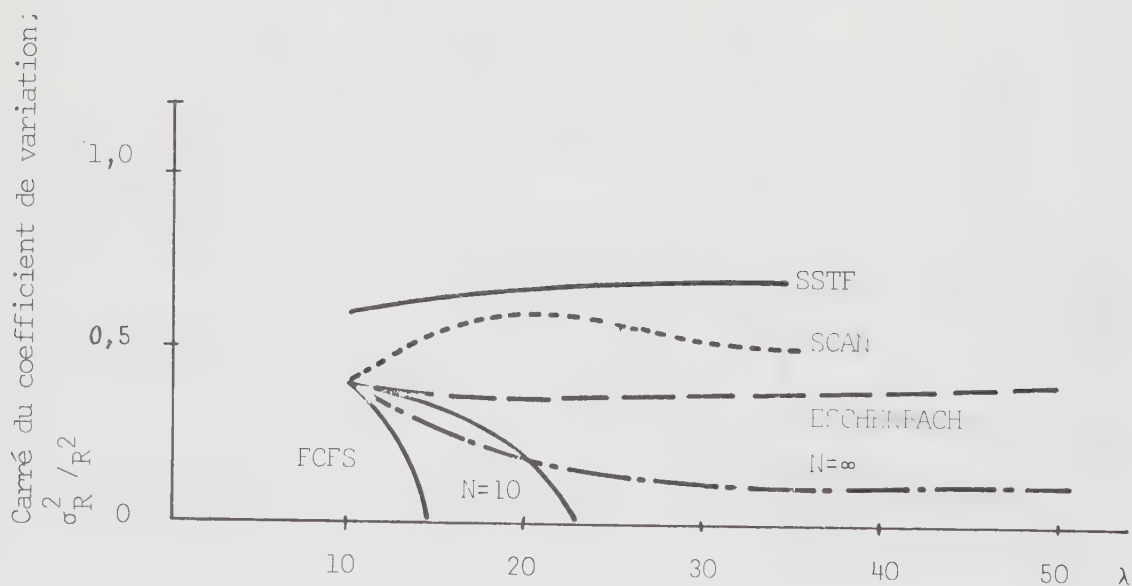


Figure 22

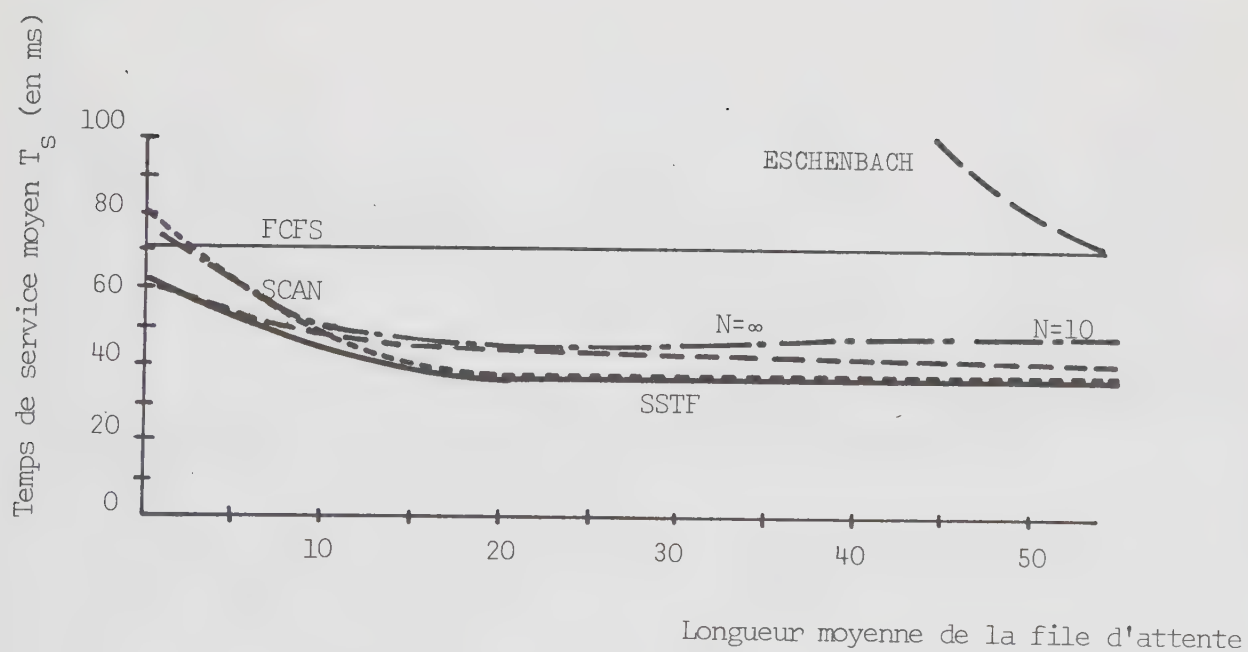


Figure 23

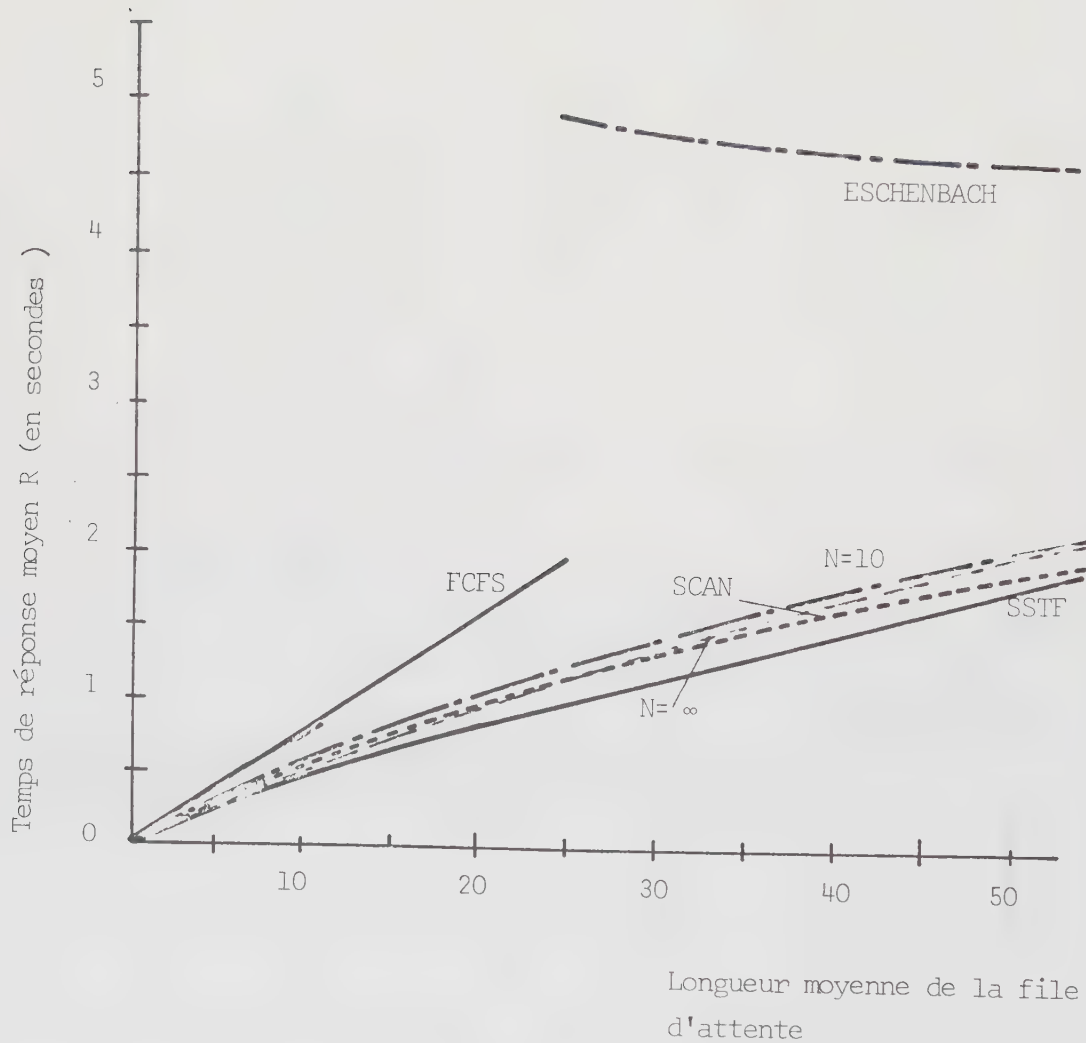


Figure 24

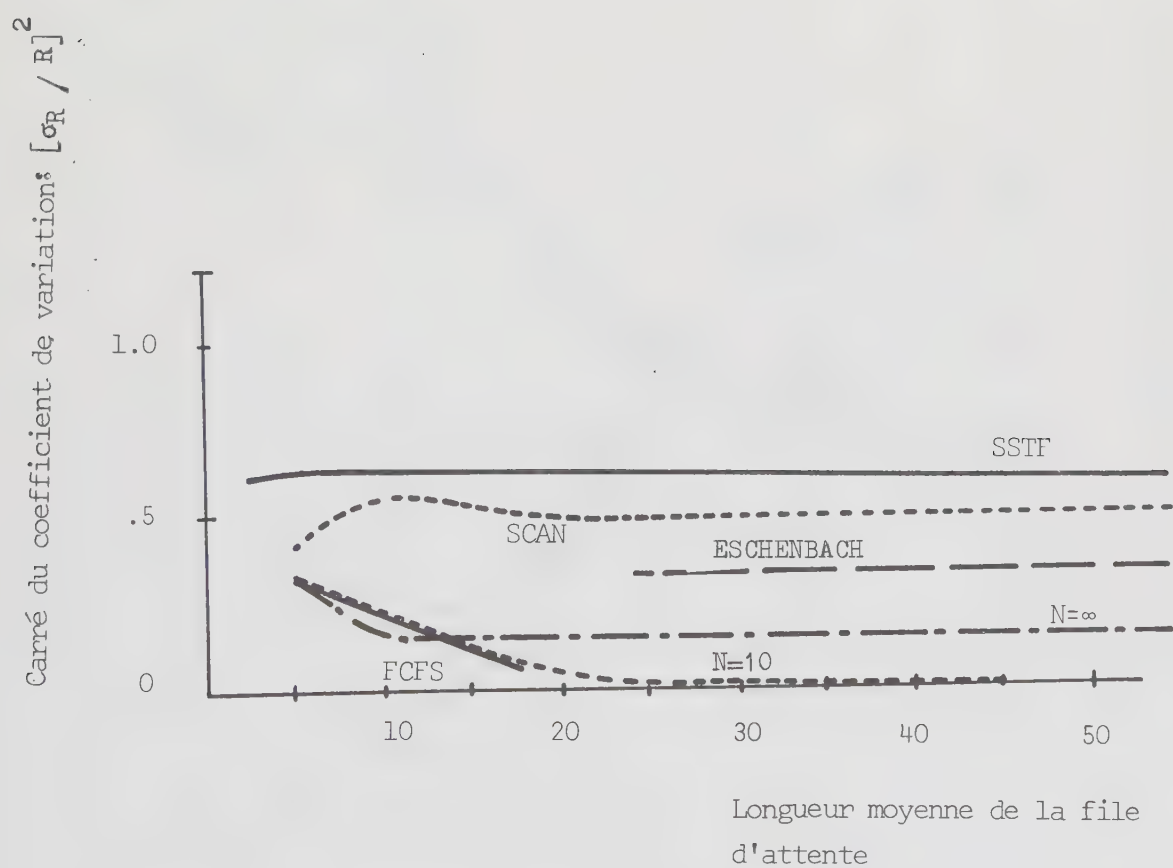
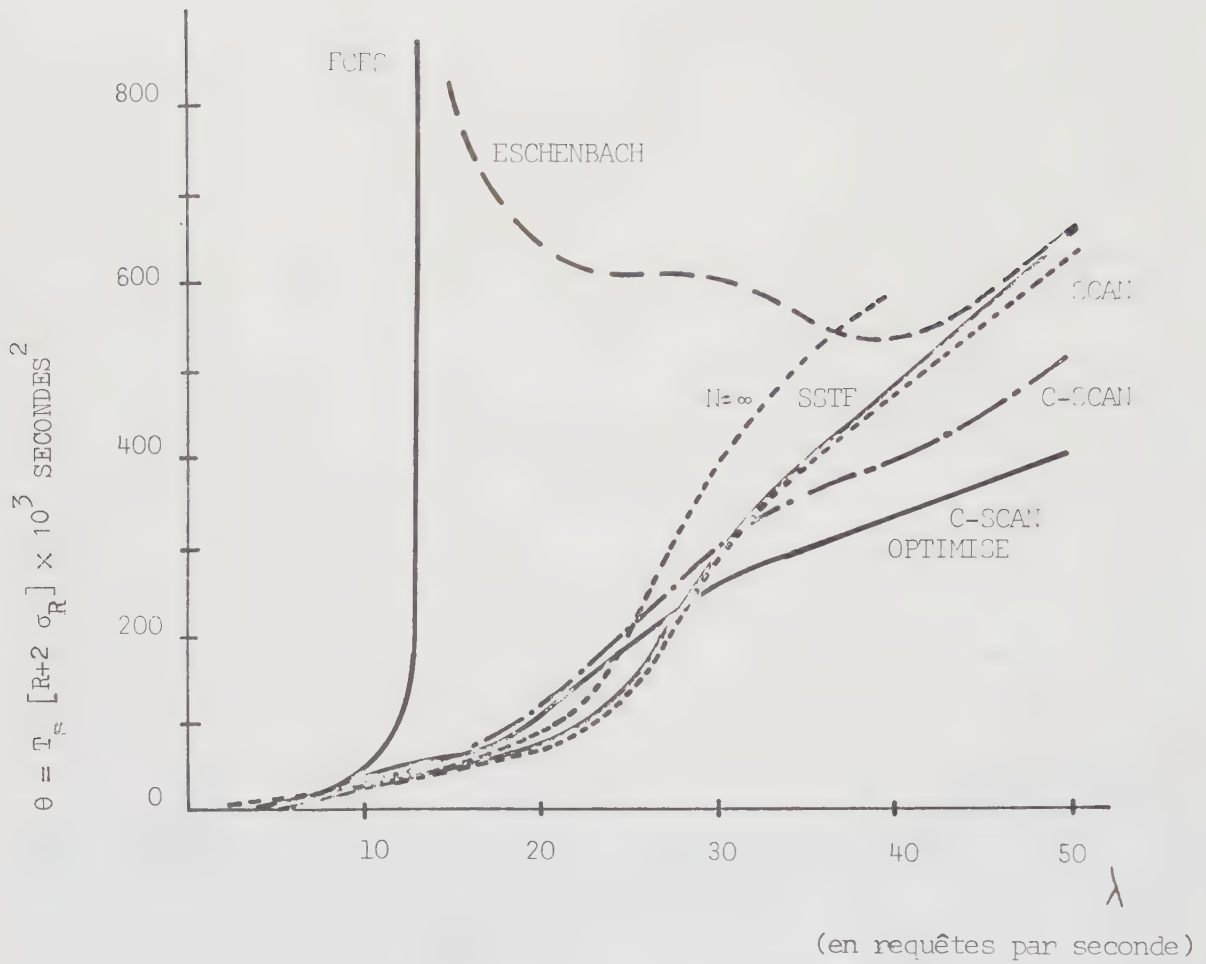
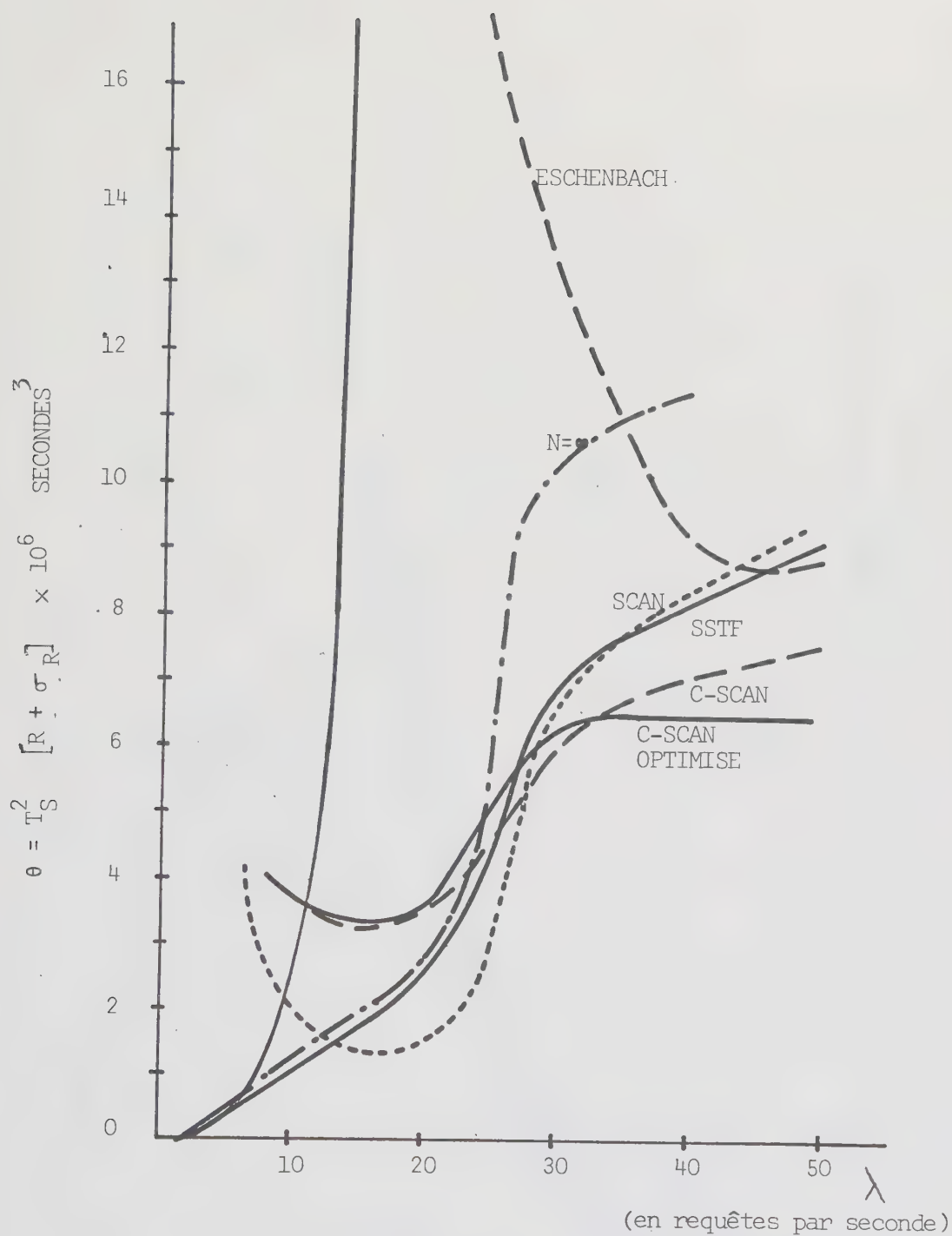


Figure 25



Performance globale
(4 secteurs par piste)

Figure 26



Performance globale (4 secteurs par piste)

Figure 27

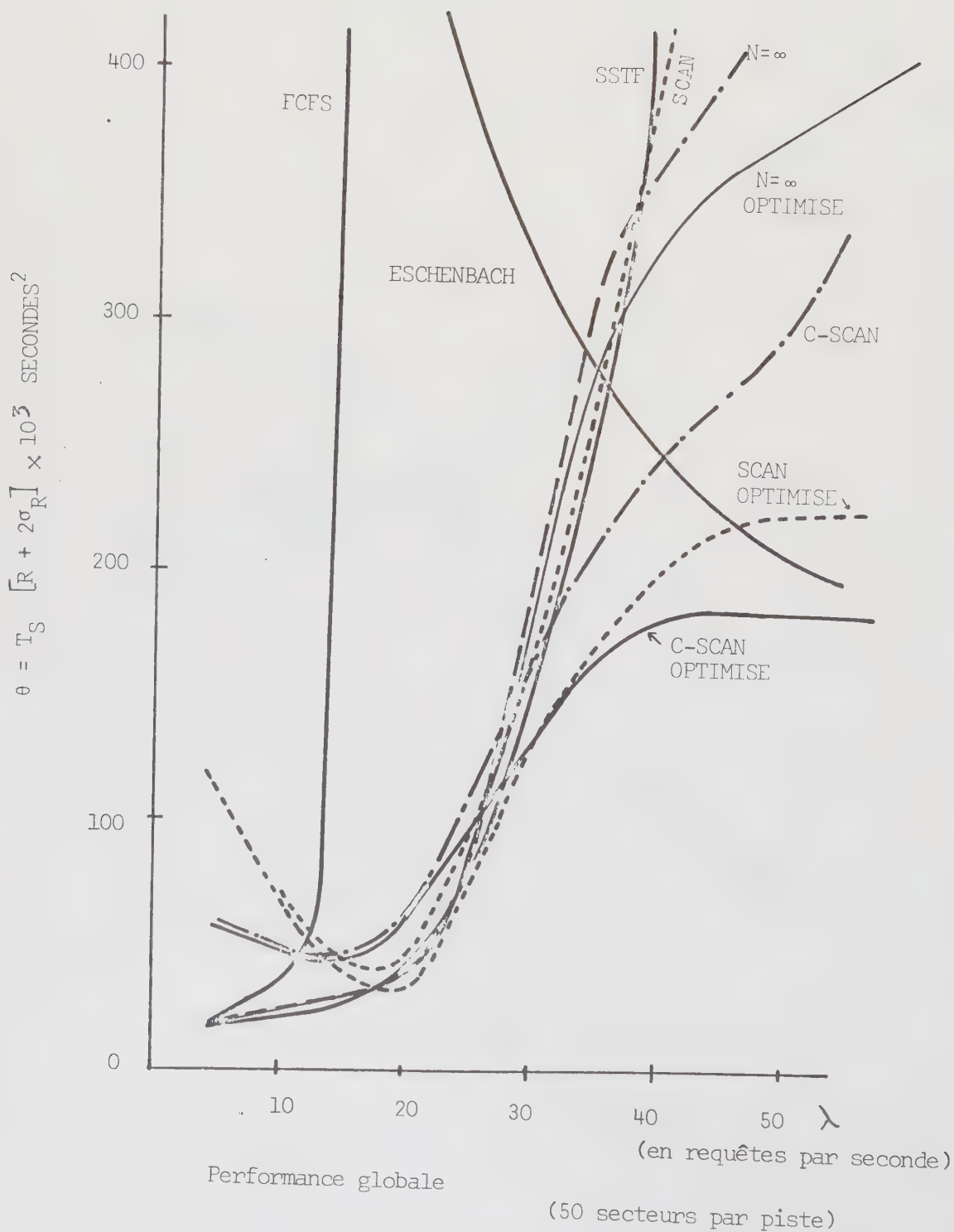


Figure 28

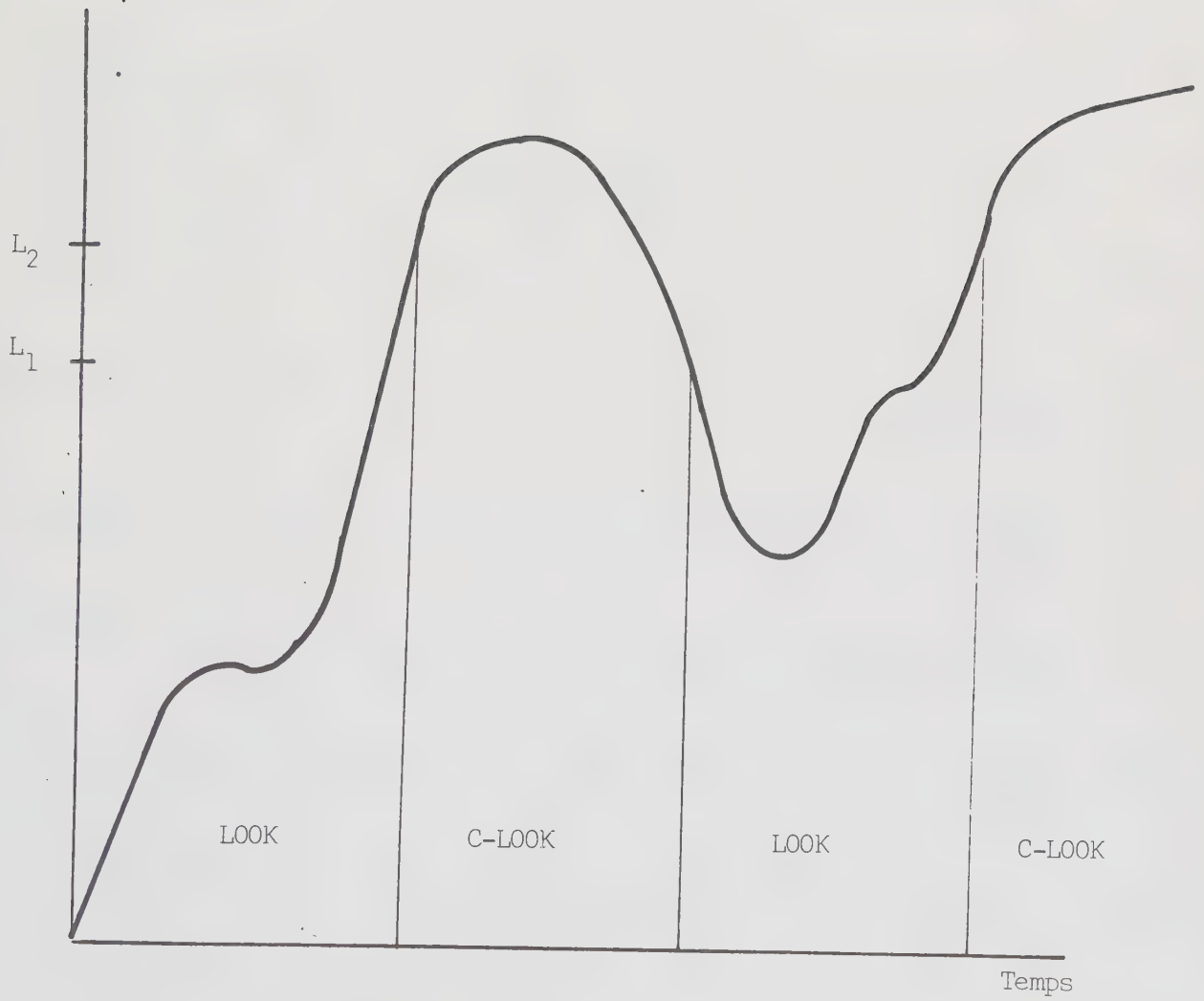


Figure 29

REFERENCES

- [1] WEINGARTEN A. : The Eschenbach drum scheme - CACM vol 9, n° 7 (juillet 1966)
- [2] COFFMAN, E.G. : Analysis of a drum Input/Output queue under scheduled operation in a paged computer system - JACM vol. 16, n° 1
- [3] BURGE, W.H., KONHEIM, A.G. : An accessing model - JACM vol. 13 n° 3 (Juillet 1971)
- [4] FULLER, S.H. : An optimal drum scheduling algorithm - IEEE Transactions on Computers, vol. C-21, n° 11 (novembre 1972)
- [5] TEOREY, T.J., PINKERTON, T.B. : A comparative analysis of disk scheduling policies - CACM vol. 15, n° 3 (mars 1972)
- [6] DENNING, P.J. : Effects of scheduling on file memory operations. Proc. of AFIPS SJCC (1967)
- [7] FRANK, H. : Analysis and optimisation of disk storage devices for time-sharing systems - JACM vol. 16, n° 4 (octobre 1969)
- [8] ABATE, J., DUBNER, H., WEINBERG, S.B. : Queuing analysis of the IBM 2314 disk storage facility - JACM vol 15, n° 4 (octobre 1968)
- [9] TAKACS : Introduction to the theory of queues - Oxford University Press (pages 66-67)
- [10] LITTLE, J.D.C. : A proof for the queuing formula : $L = \lambda W$ - OPERATIONS RESEARCH vol. 9 n° 3 (mai-juin 1961)
- [11] JEWELL, W.S. : A simple proof of $L = \lambda W$ - OPERATIONS RESEARCH vol. 15
- [12] TEOREY, T.J. : Properties of disk scheduling policies in multiprogrammed computer systems - Proc. of AFIPS FJCC (1972)
- [13] MANOCHA, T. : Ordered motion for direct-access devices - SIAI 1971 Fall meeting Madison Wisconsin (1971)

- [14] COFFMAN, E.G., TURNBULL, C.J.M. : A note on the relative performance of two disk scanning policies - I.P.L. vol 2
- [15] COFFMAN, E.G., KLIMKO, L.A., RYAN : Analysis of scanning policies for reducing disk seek times. SIAM J. on Computing
- [16] GROSSMAN, D.D., SILVERMAN, H.F. : Placement of records on a secondary storage device to minimise access time - JACM vol. 20 n° 3 (juillet 1973)
- [17] GELENBE, E., LENFANT, J., POTIER, D. : Response time of a fixed head disk to transfers of variable lenght - SIAM J. on Comp., à paraître.

